

# A Parallelizable Dual Smoothing Method for Large Scale Convex Regression Problems

Necdet Serhat Aybat · Zi Wang

Date: 08/07/2016

**Abstract** Convex regression (CR) is an approach for fitting a convex function to a finite number of observations. It arises in various applications from diverse fields such as statistics, operations research, economics, and electrical engineering. The least squares (LS) estimator, which can be computed via solving a quadratic program (QP), is an intuitive method for convex regression with already established strong theoretical guarantees. On the other hand, since the number of constraints in the QP formulation increases quadratically in the number of observed data points, the QP quickly becomes impractical to solve using traditional interior point methods. To address this issue, we propose a first-order method based on dual smoothing that carefully manages the memory usage through parallelization in order to efficiently compute the LS estimator in practice for large-scale CR instances.

**Keywords** Convex regression · Tikhonov regularization · Dual smoothing · Parallel method · First-order method · active set method · ADMM

## 1 Introduction

*Convex regression* (CR) problem deals with fitting a convex function to a given finite set of location/observation pairs, where each pair consists of a vector of independent variables and corresponding scalar dependent variable. In particular, suppose  $N$  location/observation pairs are given  $\{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell=1}^N \subset \mathbb{R}^n \times \mathbb{R}$  satisfying

$$\bar{y}_\ell = f_0(\bar{x}_\ell) + \varepsilon_\ell, \quad \ell = 1, \dots, N, \quad (1)$$

A preliminary version of this work [4] has appeared in the Proceedings of the IEEE Conference on Decision and Control.

---

N. S. Aybat  
Industrial and Manufacturing Engineering Dept., Penn State University  
University Park, PA 16802, USA E-mail: nsa10@psu.edu

Z. Wang  
Industrial and Manufacturing Engineering Dept., Penn State University  
University Park, PA 16802, USA E-mail: zzw121@psu.edu

where  $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$  is a convex function, and  $\varepsilon_\ell$  is a random noise with  $E[\varepsilon_\ell] = 0$  for all  $\ell$ . The objective is to infer the convex function  $f_0$  from the noisy observations  $\{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell=1}^N$ . CR problems arise in various applications coming from diverse fields such as statistics, operations research, economics, and electrical engineering. M. Mousavi [28] used CR to estimate the value function for Markov chains with expected infinite-horizon discounted rewards, which naturally arises in various control problems, and estimating value functions is essential for approximate dynamic programming and applied probability. In economics, CR has been adopted for approximating consumers' concave utility functions from empirical data [26]. Moreover, in queueing network context, when the expectation of a performance measure is convex in model parameters – see [10], then using Monte Carlo methods to compute the expectation gives rise to a CR problem [22].

CR was first studied in [19] for estimating concave production functions. Later, various solution methods were proposed in the uni-variate setting, e.g., [11, 7, 36]. The problem of fitting a convex function in the multi-variate setting has been considered in [20, 21] where the minimization of the least squares (LS) error subject to the first-order convexity shape constraints is studied; furthermore, [1, 2] also considered the same approach with additional second-order convexity constraints. The most well-known method for CR is to solve the LS problem,

$$\hat{f}_N = \arg \min_{f \in \mathcal{C}} \sum_{\ell=1}^N (f(\bar{x}_\ell) - \bar{y}_\ell)^2, \quad (2)$$

where  $\mathcal{C} \triangleq \{f : \mathbb{R}^n \rightarrow \mathbb{R} \text{ such that } f \text{ is convex}\}$ . This *infinite* dimensional problem is equivalent to a finite dimensional quadratic problem (QP) given in (3) – see Proposition 1 in [22],

$$\min_{y_\ell \in \mathbb{R}, \xi_\ell \in \mathbb{R}^n, \ell=1, \dots, N} \left\{ \sum_{\ell=1}^N |y_\ell - \bar{y}_\ell|^2 : y_{\ell_2} - y_{\ell_1} + \xi_{\ell_1}^\top (\bar{x}_{\ell_1} - \bar{x}_{\ell_2}) \geq 0, \quad 1 \leq \ell_1 \neq \ell_2 \leq N \right\}. \quad (3)$$

Indeed, let  $\{(y_\ell^*, \xi_\ell^*)\}_{\ell=1}^N$  be an optimal solution to (3), it is easy to show that when  $N \geq n + 1$ ,  $\{y_\ell^*\}_{\ell=1}^N$  is unique,  $\hat{f}_N(\bar{x}_\ell) = y_\ell^*$  and  $\xi_\ell^* \in \partial \hat{f}_N(\bar{x}_\ell)$  for all  $\ell$ , where  $\partial$  denotes the subdifferential operator. The theoretical behavior of the LS estimator has been studied thoroughly in the past 50 years. In the *univariate* setting, i.e.,  $n = 1$ , the consistency of the LS estimator is proved in [18]; and the convergence rate of the estimator is established in [24]. Groeneboom et al. [14] extended these results and derived the asymptotic distribution of LS estimator at a fixed point of positive curvature. In the *multivariate* setting, the consistency is shown in [22], i.e.,  $\hat{f}_N \rightarrow f_0$  almost surely as  $N$  increases.

Besides LS estimator, there are other methods for solving CR problem in the multivariate setting. A heuristic approach is proposed in [23] to compute locally optimal fits, which has no convergence guarantee. A convex adaptive partitioning (CAP) method is proposed in [15], which creates a globally convex regression model via computing locally linear fits on adaptively selected co-variate partitions. Both methods use the piecewise linear model, and minimize the least square error. In addition, more recently, Hannah and Dunson [16]

proposed a new estimator based on using traditional ensemble methods to average over multiple piecewise linear estimators, and proved its consistency when CAP is the underlying estimator. However, LS estimator has some significant advantages over the methods mentioned above. First, LS estimator is a non-parametric regression method as discussed in [35], which does not require any tuning parameters and avoids the issue of selecting an appropriate estimation structure; however, as also pointed out in [28], the methods proposed in [15, 17] are semi-parametric, and require adjusting several parameters before fitting a convex function. Second, LS estimator can be computed by solving the QP in (3); therefore, at least in theory, it can be solved very efficiently using interior point methods (IPM). A major drawback of the LS estimator in practice is that the number of shape constraints in (3) is  $\mathcal{O}(N^2)$ . Consequently, the problem quickly becomes massive even for moderate number of observations: for off-the-shelf IPMs that do not exploit any structural properties of (3), the complexity of each factorization step is  $\mathcal{O}(N^3(n+1)^3)$ , and the memory requirement is  $\mathcal{O}(N^2(n+1)^2)$  assuming Cholesky factors are stored - see [8, 32] - for more detailed discussion on memory usage and computational complexity of both IPM and our proposed method (exploiting the structure), see Section 2.5.

In this paper, we propose a new *parallelizable* method for computing the LS estimator on large-scale CR problems. The proposed method can efficiently solve large-scale instances of (3) by carefully managing the memory usage through parallelization, and exploiting the underlying problem structure. In particular, the proposed method, P-APG, is based on dual smoothing, i.e., regularizing the objective in (3) with a strongly convex function. More specifically, we adopted Tikhonov regularization, which leads to a differentiable dual function with a Lipschitz continuous gradient. Compared to the traditional dual decomposition methods, the dual smoothing based approaches can guarantee feasibility of primal iterate sequence in the limit. To briefly summarize, P-APG is an iterative method to solve the regularized QP problem in (7) through solving a number of small-size QPs in each iteration. In our main results, Theorem 2 and 3, we establish error bounds on the quality of inexact solutions to the regularized problem; particularly, we investigate how well the inexact solutions can approximate i) function values of the LS estimator, i.e.,  $\hat{f}_N(\bar{x}_\ell)$ , and ii) subgradients from the subdifferential of the LS estimator, i.e.,  $\partial \hat{f}_N(\bar{x}_\ell)$ . Next, we study the convergence behavior of P-APG to compute these function value and subgradient approximations. In Section 2.4.1, we show that using a *continuation* method, we can construct an iterate sequence that is asymptotically optimal to the original LS problem in (3) with a provable convergence rate. We adopted a primal-dual IPM to solve the small-size QP subproblems arising in P-APG iterations, and analyzed the computational complexity of an P-APG iteration by exploiting the special structure of the constraints and the objective function. In the rest, as alternatives to P-APG, we examined how an active set method (ASM) can be efficiently implemented to solve (3), and briefly discussed a recently proposed ADMM algorithm [25] for (3). Finally, we conclude with a number of numerical examples comparing P-APG, ASM, and ADMM. Our results show that P-APG is the method of choice for large  $N$ .

*Notations:* Throughout, i.i.d. is short for independent and identically distributed.  $\mathbf{I}_n$  denotes the  $n \times n$ -identity matrix. Given  $x \in \mathbb{R}^n$ ,  $(x)_+ \triangleq \max\{x, 0\}$  and  $(x)_- \triangleq \min\{x, 0\}$ ; hence,  $x = (x)_+ + (x)_-$ . For  $x, y \in \mathbb{R}^n$ ,  $\langle x, y \rangle \triangleq x^\top y$  represents the standard inner product.  $\mathbf{1}$  denotes the vector of all ones, and  $e_i \in \mathbb{R}^n$  denotes the  $i$ -th unit vector for each  $i \in \{1, \dots, n\}$ .

## 2 Methodology

Let  $f_0 : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$  be the *unknown* proper convex function generating the observed data  $\{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell=1}^N \subset \mathbb{R}^n \times \mathbb{R}$  as in (1), and let  $\mathcal{N} := \{1, \dots, N\}$  denote the set of indices corresponding to  $N$  observations. Suppose  $B_x > 0$  such that  $\|\bar{x}_\ell\|_2 \leq B_x$  for all  $\ell \in \mathcal{N}$ . Define the long-vector notations for the variables:  $\mathbf{y} = [y_\ell]_{\ell \in \mathcal{N}} \in \mathbb{R}^N$ , and  $\boldsymbol{\xi} = [\xi_\ell]_{\ell \in \mathcal{N}} \in \mathbb{R}^{Nn}$ .

Consider (3) in the following compact form:

$$\chi^* \triangleq \arg \min_{\mathbf{y} \in \mathbb{R}^N, \boldsymbol{\xi} \in \mathbb{R}^{Nn}} \left\{ \frac{1}{2} \|\mathbf{y} - \bar{\mathbf{y}}\|_2^2 : A_1 \mathbf{y} + A_2 \boldsymbol{\xi} \geq 0 \right\}, \quad (4)$$

where  $A_1 \in \mathbb{R}^{N(N-1) \times N}$  and  $A_2 \in \mathbb{R}^{N(N-1) \times Nn}$  are the matrices corresponding to constraints in (3). Let  $(\mathbf{y}^*, \boldsymbol{\xi}^*)$  be the least-norm optimal solution in  $\chi^*$ , i.e.,

$$(\mathbf{y}^*, \boldsymbol{\xi}^*) \triangleq \arg \min_{\mathbf{y}, \boldsymbol{\xi}} \left\{ \frac{1}{2} \|\mathbf{y}\|_2^2 + \frac{1}{2} \|\boldsymbol{\xi}\|_2^2 : (\mathbf{y}, \boldsymbol{\xi}) \in \chi^* \right\}. \quad (5)$$

It is easy to show that  $\mathbf{y}^*$  is unique to (4), i.e., if  $(\mathbf{y}, \boldsymbol{\xi}) \in \chi^*$ , then  $\mathbf{y} = \mathbf{y}^*$  – see Proposition 1 in [22]. Hence, it follows from (5) that  $\boldsymbol{\xi}^*$  has the least norm, i.e., for all  $(\mathbf{y}, \boldsymbol{\xi}) \in \chi^*$ , one has  $\|\boldsymbol{\xi}\|_2 \geq \|\boldsymbol{\xi}^*\|_2$ . Moreover, since (4) is a convex QP, strong duality holds, and an optimal dual solution  $\boldsymbol{\theta}^* \in \mathbb{R}^{N(N-1)}$  exists.

Note for each  $(\ell_1, \ell_2) \in \mathcal{P} \triangleq \{(\ell_1, \ell_2) \in \mathcal{N} \times \mathcal{N} : \ell_1 \neq \ell_2\}$ , there is a constraint in (3), i.e.,  $y_{\ell_2} - y_{\ell_1} + \xi_{\ell_1}^\top (\bar{x}_{\ell_1} - \bar{x}_{\ell_2}) \geq 0$  corresponds to  $(\ell_1, \ell_2) \in \mathcal{P}$ . In order to fix  $A_1$  and  $A_2$ , we sort the rows according to increasing lexicographic order on the index set  $\mathcal{P}$ , i.e., the row for the constraint corresponding to  $(\ell_1, \ell_2)$  comes before than the one corresponding to  $(\ell_3, \ell_4)$  if either  $\ell_1 < \ell_3$ , or  $\ell_1 = \ell_3$  and  $\ell_2 < \ell_4$ . Next, we give explicit forms for  $A_1$  and  $A_2$ .

**Definition 1** Let  $T_\ell \in \mathbb{R}^{N-1 \times N}$  such that  $T_\ell = [e_1 \cdots e_{\ell-1} \quad -\mathbf{1} \quad e_\ell \cdots e_{N-1}]$  for  $\ell \in \mathcal{N}$ , where  $e_j \in \mathbb{R}^{N-1}$  is the  $j$ -th unit vector for  $j \in \{1, \dots, N-1\}$ . Moreover, let  $\bar{X} \in \mathbb{R}^{N \times n}$  such that  $\bar{X} = [\bar{x}_\ell^\top]_{\ell \in \mathcal{N}}$ , i.e.,  $\{\bar{x}_\ell\}_{\ell \in \mathcal{N}}$  are the rows of  $\bar{X}$ . Then  $A_1 = [T_\ell]_{\ell \in \mathcal{N}}$ , obtained by vertically concatenating  $\{T_\ell\}_{\ell \in \mathcal{N}}$ , and  $A_2 = \text{diag} \{-T_\ell \bar{X}\}_{\ell \in \mathcal{N}}$  is a block-diagonal matrix as given below

$$A_1 = \begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ T_N \end{pmatrix}, \quad A_2 = \begin{pmatrix} X_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & X_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & X_N \end{pmatrix}, \quad X_\ell \triangleq -T_\ell \bar{X}, \quad \ell \in \mathcal{N}. \quad (6)$$

## 2.1 Separability

Given the regularization parameter  $\gamma \geq 0$ , consider

$$(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*) \triangleq \arg \min_{\mathbf{y}, \boldsymbol{\xi}} \left\{ r_\gamma(\mathbf{y}, \boldsymbol{\xi}) \triangleq \frac{1}{2} \|\mathbf{y} - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}\|_2^2 : A_1 \mathbf{y} + A_2 \boldsymbol{\xi} \geq 0 \right\}. \quad (7)$$

Note simply setting  $\gamma = 0$  in (7), we obtain the original problem (4).

To reduce the *curse of dimensionality* and develop a *parallelizable* method that can solve problems in (4) and (7) for large  $N$ , we employ dual decomposition to induce *separability*. To this aim, we *partition* the observation set into  $K$  subsets  $\{\mathcal{C}_i\}_{i \in \mathcal{K}}$ , where  $\mathcal{K} \triangleq \{1, \dots, K\}$  denote the set of indices corresponding to  $K$  subsets of  $\mathcal{N}$ . In particular, we choose  $\{\mathcal{C}_i\}_{i \in \mathcal{K}}$  as a partition of  $\mathcal{N}$  such that  $|\mathcal{C}_i| \geq n + 1$  for all  $i$ . To simplify the notation, throughout the paper we make the following assumption.

**Assumption 1** Suppose  $N = K\bar{N}$  for some  $\bar{N} > n + 1$ , and without loss of generality assume that  $\mathcal{C}_i \triangleq \{(i-1)\bar{N} + 1, (i-1)\bar{N} + 2, \dots, i\bar{N}\}$  for  $i \in \mathcal{K}$ .

Throughout the paper, for each  $i \in \mathcal{K}$ , let  $\mathbf{y}_i \in \mathbb{R}^{\bar{N}}$  and  $\boldsymbol{\xi}_i \in \mathbb{R}^{\bar{N}n}$  denote the sub-vectors of  $\mathbf{y} \in \mathbb{R}^N$  and  $\boldsymbol{\xi} \in \mathbb{R}^{Nn}$  corresponding to indices in  $\mathcal{C}_i$ , respectively. In particular, for all  $i \in \mathcal{K}$ ,  $\mathbf{y}_i = [y_\ell]_{\ell \in \mathcal{C}_i}$  and  $\boldsymbol{\xi}_i = [\xi_\ell]_{\ell \in \mathcal{C}_i}$ . Similarly, we define the same long-vectors for the observation data:  $\bar{\mathbf{y}}_i = [\bar{y}_\ell]_{\ell \in \mathcal{C}_i} \in \mathbb{R}^{\bar{N}}$ .

**Definition 2** Define  $\mathcal{P} \triangleq \{(\ell_1, \ell_2) \in \mathcal{N} \times \mathcal{N} : \ell_1 \neq \ell_2\}$  and  $\mathcal{G} \triangleq \{(i, j) \in \mathcal{K} \times \mathcal{K} : i \neq j\}$ . For each  $i \in \mathcal{K}$ , let  $A_1^{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1) \times \bar{N}}$  and  $A_2^{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1) \times \bar{N}n}$  be the submatrices of  $A_1$  and  $A_2$  such that they consist of the rows corresponding to row indices  $(\ell_1, \ell_2) \in \mathcal{P}$  for  $\ell_1, \ell_2 \in \mathcal{C}_i$ . Similarly, for each  $(i, j) \in \mathcal{G}$ , let  $A_1^{ij} \in \mathbb{R}^{\bar{N}^2 \times \bar{N}}$  and  $A_2^{ij} \in \mathbb{R}^{\bar{N}^2 \times \bar{N}n}$  be the submatrices of  $A_1$  and  $A_2$  consisting of the rows corresponding to indices  $\{(\ell_1, \ell_2) \in \mathcal{P} : \ell_1 \in \mathcal{C}_i, \ell_2 \in \mathcal{C}_j\}$ .

Furthermore, for each  $i \in \mathcal{K}$ , let  $\bar{A}_1^{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1) \times \bar{N}}$  and  $\bar{A}_2^{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1) \times \bar{N}n}$  be the submatrices of  $A_1^{ii}$  and  $A_2^{ii}$  such that  $\bar{A}_1^{ii}$  consists of the columns of  $A_1^{ii}$  corresponding to  $\mathbf{y}_i$ ; and  $\bar{A}_2^{ii}$  consists of the columns of  $A_2^{ii}$  corresponding to  $\boldsymbol{\xi}_i$ .

Note that for every ordered pair  $(\ell_1, \ell_2) \in \mathcal{P}$ , there corresponds a constraint in (3), which is represented by a row in matrices  $A_1$  and  $A_2$  of formulations (4) and (7). Consider all the constraints in (3) corresponding to those pairs  $(\ell_1, \ell_2)$  such that they belong to different sets in the partition, i.e.,  $\ell_1 \in \mathcal{C}_i$ ,  $\ell_2 \in \mathcal{C}_j$  for some  $(i, j) \in \mathcal{G}$ , let  $\boldsymbol{\theta}_{ij} \in \mathbb{R}^{\bar{N}^2}$  denote the associated dual variables, and  $\boldsymbol{\theta} = [\boldsymbol{\theta}_{ij}]_{(i,j) \in \mathcal{G}} \in \mathbb{R}^{\bar{N}^2 K(K-1)}$  denote the vector formed by vertically concatenating  $\boldsymbol{\theta}_{ij}$  for  $1 \leq i \neq j \leq K$ . By dualizing all such constraints in (7), we form the *partial* Lagrangian function:

$$\mathcal{L}_\gamma(\mathbf{y}, \boldsymbol{\xi}, \boldsymbol{\theta}) \triangleq \frac{1}{2} \sum_{i \in \mathcal{K}} \left( \|\mathbf{y}_i - \bar{\mathbf{y}}_i\|_2^2 + \gamma \|\boldsymbol{\xi}_i\|_2^2 \right) - \sum_{(i,j) \in \mathcal{G}} \left\langle \boldsymbol{\theta}_{ij}, A_1^{ij} \mathbf{y} + A_2^{ij} \boldsymbol{\xi} \right\rangle. \quad (8)$$

and obtain the following partial dual function

$$g_\gamma(\boldsymbol{\theta}) \triangleq \min_{\mathbf{y}, \boldsymbol{\xi}} \left\{ \mathcal{L}_\gamma(\mathbf{y}, \boldsymbol{\xi}, \boldsymbol{\theta}) : A_1^{ii} \mathbf{y} + A_2^{ii} \boldsymbol{\xi} \geq 0, i \in \mathcal{K} \right\}. \quad (9)$$

Hence, the dual problem corresponding to (7) is given as

$$\Theta_\gamma^* \triangleq \arg \max \{g_\gamma(\boldsymbol{\theta}) : \boldsymbol{\theta} \geq 0\}, \quad \text{and} \quad p_\gamma^* \triangleq g_\gamma(\boldsymbol{\theta}_\gamma^*) \quad \text{for} \quad \boldsymbol{\theta}_\gamma^* \in \Theta_\gamma^*. \quad (10)$$

Since strong-duality trivially holds between the primal-dual problem pair, (7) and (10), we have

$$p_\gamma^* = r_\gamma(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*) = \frac{1}{2} \|\mathbf{y}_\gamma^* - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}_\gamma^*\|_2^2. \quad (11)$$

For any given regularization parameter  $\gamma \geq 0$  and dual variable  $\boldsymbol{\theta}$ , the partial Lagrangian function  $\mathcal{L}_\gamma$  is *separable* in  $\{(\mathbf{y}_i, \boldsymbol{\xi}_i)\}_{i \in \mathcal{K}}$ , and can be written as

$$\mathcal{L}_\gamma(\mathbf{y}, \boldsymbol{\xi}, \boldsymbol{\theta}) = \sum_{i \in \mathcal{K}} \mathcal{L}_\gamma^i(\mathbf{y}_i, \boldsymbol{\xi}_i, \boldsymbol{\theta}) \quad (12)$$

for some very simple quadratic function,  $\mathcal{L}_\gamma^i$ , of  $(\mathbf{y}_i, \boldsymbol{\xi}_i)$  for each  $i \in \mathcal{K}$ . Moreover, after partially dualizing some of the constraints as shown in (8), the remaining ones in (9) define a superset,  $\mathcal{Q}$ , of the original feasible region. Indeed,  $\mathcal{Q} = \{(\mathbf{y}, \boldsymbol{\xi}) : A_1^{ii} \mathbf{y} + A_2^{ii} \boldsymbol{\xi} \geq 0, i \in \mathcal{K}\} = \{(\mathbf{y}, \boldsymbol{\xi}) : A_1^{ii} \mathbf{y}_i + \bar{A}_2^{ii} \boldsymbol{\xi}_i \geq 0, i \in \mathcal{K}\}$  – since the entries of  $A_1^{ii}$  that does not belong to its submatrix  $\bar{A}_1^{ii}$  are all 0; and similarly, the entries of  $A_2^{ii}$  that does not belong to its submatrix  $\bar{A}_2^{ii}$  are all 0 as well. Therefore, we have  $\mathcal{Q} = \bigotimes_{i \in \mathcal{K}} \mathcal{Q}_i$ , where  $\mathcal{Q}_i \triangleq \{(\mathbf{y}_i, \boldsymbol{\xi}_i) : A_1^{ii} \mathbf{y}_i + \bar{A}_2^{ii} \boldsymbol{\xi}_i \geq 0\}$  for  $i \in \mathcal{K}$ , and  $\bigotimes$  denotes the Cartesian product. Consequently, since  $\mathcal{L}_\gamma$  is separable as shown in (12), computing the partial dual function  $g_\gamma(\boldsymbol{\theta})$  in (9) is equivalent to solving  $K$  quadratic subproblems, i.e., one for each  $i \in \mathcal{K}$ ,

$$\min_{\mathbf{y}_i \in \mathbb{R}^N, \boldsymbol{\xi}_i \in \mathbb{R}^{N_n}} \left\{ \mathcal{L}_\gamma^i(\mathbf{y}_i, \boldsymbol{\xi}_i, \boldsymbol{\theta}) : \bar{A}_1^{ii} \mathbf{y}_i + \bar{A}_2^{ii} \boldsymbol{\xi}_i \geq 0 \right\}. \quad (13)$$

Given the dual variables  $\boldsymbol{\theta}$ , since all  $K$  subproblems can be computed in *parallel*, one can take advantage of the computing power of multi-core processors. In the rest of the paper, we discuss how to compute a solution to (3) via solving the dual problem:  $\max\{g_\gamma(\boldsymbol{\theta}) : \boldsymbol{\theta} \geq 0\}$ .

## 2.2 Projected Subgradient Method for Dual

Clearly, for  $\gamma = 0$ ,  $g_0$  defined in (9) is the dual function for the original problem (4); and the projected subgradient method can be adopted for solving the dual problem  $\max\{g_0(\boldsymbol{\theta}) : \boldsymbol{\theta} \geq 0\}$ . Let  $\boldsymbol{\theta} = \mathbf{0}$ , i.e.,  $\boldsymbol{\theta}_{ij}^0 = \mathbf{0}$  for all  $(i, j) \in \mathcal{G}$ . Given the  $k$ -th dual iterate  $\boldsymbol{\theta}^k$ , let  $(\mathbf{y}^k, \boldsymbol{\xi}^k)$  denote an optimal solution to the minimization problem in (9) when  $\gamma = 0$  and  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}^k$ ; and let  $\boldsymbol{\theta}_{ii}^k$  denote an optimal dual associated with constraints  $A_1^{ii} \mathbf{y} + A_2^{ii} \boldsymbol{\xi} \geq 0$  in (9). The next dual iterate  $\boldsymbol{\theta}^{k+1}$  is computed for an appropriately chosen step size  $t_k > 0$ :

$$\boldsymbol{\theta}_{ij}^{k+1} = \Pi_{\mathcal{S}_{ij}^k} \left( \boldsymbol{\theta}_{ij}^k - t_k \left( A_1^{ij} \mathbf{y}^k + A_2^{ij} \boldsymbol{\xi}^k \right) \right), \quad (14)$$

where  $\Pi_{\mathcal{S}_{ij}^k}(\cdot)$  denotes the Euclidean projection on to

$$\mathcal{S}_{ij}^k \triangleq \left\{ \boldsymbol{\theta}_{ij} \geq \mathbf{0} : \boldsymbol{\theta}_{ij}^\top A_2^{ij} + \boldsymbol{\theta}_{ii}^k{}^\top A_2^{ii} = \mathbf{0} \right\}.$$

Since the Lagrangian function  $\mathcal{L}_0$  is linear in  $\boldsymbol{\xi}$  when  $\gamma = 0$ ,  $\text{dom } g_0$  is non-trivial; hence the projection on to the Cartesian product  $\bigotimes_{(i,j) \in \mathcal{G}} \mathcal{S}_{ij}^k$  ensures  $\boldsymbol{\theta}^{k+1} \in \text{dom } g_0$ . The projected subgradient method is guaranteed to converge in function value for a diminishing step size sequence  $\{t_k\}_{k=1}^\infty$ , and it requires  $\mathcal{O}(1/\epsilon^2)$  iterations to obtain an  $\epsilon$ -optimal solution –see [29]. On the other hand, even if the dual iterates converge to an optimal dual solution  $\boldsymbol{\theta}^*$ , the primal feasibility of the corresponding primal iterate sequence  $\{(\mathbf{y}^k, \boldsymbol{\xi}^k)\}$  cannot be guaranteed in the limit as it might converge to a stationary point of the Lagrangian  $\mathcal{L}_0(\cdot, \cdot, \boldsymbol{\theta}^*)$  that is primal infeasible, mainly due to lack of strict convexity, jointly in  $(\mathbf{y}, \boldsymbol{\xi})$ , of the objective in (4).

### 2.3 Tikhonov Regularization Approach

In order to ensure feasibility in the limit, which cannot be guaranteed by the subgradient method discussed above, we employ Tikhonov regularization as in (7) for  $\gamma > 0$ , of which convergence properties in general were investigated in [12]. In particular, as  $\gamma$  decreases to zero from above, the minimizer  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$ , as a function of  $\gamma$ , converges to  $(\mathbf{y}^*, \boldsymbol{\xi}^*) \in \chi^*$  defined in (5), i.e.,  $\boldsymbol{\xi}^*$  has the least norm among all  $(\mathbf{y}^*, \boldsymbol{\xi}) \in \chi^*$ .

**Lemma 1** *The minimizer of (7),  $\mathbf{y}_\gamma^*$ , as a function of the regularization parameter  $\gamma$ , is Hölder continuous from right at  $\gamma = 0$ . In particular,*

$$\|\mathbf{y}_\gamma^* - \mathbf{y}^*\|_2 \leq \|\boldsymbol{\xi}^*\|_2 \sqrt{\gamma}, \quad \forall \gamma \geq 0. \quad (15)$$

*Proof* Let  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  be the optimal solution to (7) and  $(\mathbf{y}^*, \boldsymbol{\xi}^*)$  be defined as in (5). Note that  $(\mathbf{y}^*, \boldsymbol{\xi}^*)$  and  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  are feasible to (7) and (4), respectively; hence, from the first-order optimality conditions of (7) and (4), we have

$$\begin{pmatrix} \mathbf{y}_\gamma^* - \bar{\mathbf{y}} \\ \gamma \boldsymbol{\xi}_\gamma^* \end{pmatrix}^\top \begin{pmatrix} \mathbf{y}^* - \mathbf{y}_\gamma^* \\ \boldsymbol{\xi}^* - \boldsymbol{\xi}_\gamma^* \end{pmatrix} \geq 0, \quad \begin{pmatrix} \mathbf{y}^* - \bar{\mathbf{y}} \\ \mathbf{0} \end{pmatrix}^\top \begin{pmatrix} \mathbf{y}_\gamma^* - \mathbf{y}^* \\ \boldsymbol{\xi}_\gamma^* - \boldsymbol{\xi}^* \end{pmatrix} \geq 0. \quad (16)$$

Moreover, since  $(\mathbf{y}^*, \boldsymbol{\xi}^*)$  and  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  are optimal to (4) and (7), respectively; we also have

$$\frac{1}{2} \|\mathbf{y}^* - \bar{\mathbf{y}}\|_2^2 \leq \frac{1}{2} \|\mathbf{y}_\gamma^* - \bar{\mathbf{y}}\|_2^2, \quad \frac{1}{2} \|\mathbf{y}_\gamma^* - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}_\gamma^*\|_2^2 \leq \frac{1}{2} \|\mathbf{y}^* - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}^*\|_2^2.$$

These two inequalities imply  $\|\boldsymbol{\xi}_\gamma^*\|_2 \leq \|\boldsymbol{\xi}^*\|_2$ . Finally, summing the two inequalities in (16) and using Cauchy-Schwarz, we obtain

$$\|\mathbf{y}_\gamma^* - \mathbf{y}^*\|_2^2 \leq \gamma \boldsymbol{\xi}_\gamma^{*\top} (\boldsymbol{\xi}^* - \boldsymbol{\xi}_\gamma^*) \leq \gamma \left( \|\boldsymbol{\xi}^*\|_2^2 - \|\boldsymbol{\xi}_\gamma^*\|_2^2 \right) \leq \gamma \|\boldsymbol{\xi}^*\|_2^2,$$

which implies the desired result.  $\square$

Since the objective function in (7) is strongly convex, jointly in  $\mathbf{y}$  and  $\boldsymbol{\xi}$ , when  $\gamma > 0$ , Danskin's theorem (see [6]) implies that  $g_\gamma$ , i.e., the Lagrangian dual function corresponding to (7), is differentiable; therefore, one can use gradient type methods to solve the corresponding dual problem  $\max\{g_\gamma(\boldsymbol{\theta}) : \boldsymbol{\theta} \geq 0\}$ . Moreover, strong convexity ensures that, one can solve the regularized primal problem in (7) by solving the associated dual problem in (10). Indeed, let  $\boldsymbol{\theta}_\gamma^*$  be an optimal solution to (10), we can recover  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  by computing the primal minimizers in (9) when the dual is set to  $\boldsymbol{\theta}_\gamma^*$ . In particular, achieving primal feasibility in the limit for the primal iterate sequence is not an issue provided that we can construct a dual iterate sequence that is asymptotically optimal to (10). We complete this section by formally stating this result.

**Theorem 1** *Let  $\gamma > 0$ , and  $\{\boldsymbol{\theta}^k\}$  be some dual sequence such that  $\boldsymbol{\theta}^k \geq 0$  for  $k \geq 1$  and  $\lim_{k \in \mathbb{Z}_+} g_\gamma(\boldsymbol{\theta}^k) = p_\gamma^*$ . Moreover, let  $(\mathbf{y}^k, \boldsymbol{\xi}^k)$  denote the unique optimal solution to the minimization problem in (9) when  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}^k$  for  $k \geq 1$ . Then  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  is the unique limit point of the primal sequence  $\{(\mathbf{y}^k, \boldsymbol{\xi}^k)\}$ . More specifically, for all  $k \geq 1$ , we have*

$$\|\mathbf{y}^k - \mathbf{y}_\gamma^*\|_2^2 + \gamma \|\boldsymbol{\xi}^k - \boldsymbol{\xi}_\gamma^*\|_2^2 \leq 2(p_\gamma^* - g_\gamma(\boldsymbol{\theta}^k)) \rightarrow 0. \quad (17)$$

*Proof* Let  $Q = \{(\mathbf{y}, \boldsymbol{\xi}) : A_1^{ii} \mathbf{y} + A_2^{ii} \boldsymbol{\xi} \geq 0, \quad i \in \mathcal{K}\}$ . Given  $\boldsymbol{\theta}^k \geq 0$  for any  $k \geq 1$ , since  $\mathcal{L}_\gamma(\mathbf{y}, \boldsymbol{\xi}, \boldsymbol{\theta}^k)$  is a quadratic function in  $(\mathbf{y}, \boldsymbol{\xi})$ , we can compute  $\mathcal{L}_\gamma(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*, \boldsymbol{\theta}^k)$  by using second-order Taylor expansion of around  $(\mathbf{y}^k, \boldsymbol{\xi}^k)$ :

$$\begin{aligned} \mathcal{L}_\gamma(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*, \boldsymbol{\theta}^k) &= \\ \mathcal{L}_\gamma(\mathbf{y}^k, \boldsymbol{\xi}^k, \boldsymbol{\theta}^k) &+ \begin{pmatrix} \nabla_{\mathbf{y}} \mathcal{L}_\gamma(\mathbf{y}^k, \boldsymbol{\xi}^k, \boldsymbol{\theta}^k) \\ \nabla_{\boldsymbol{\xi}} \mathcal{L}_\gamma(\mathbf{y}^k, \boldsymbol{\xi}^k, \boldsymbol{\theta}^k) \end{pmatrix}^\top \begin{pmatrix} \mathbf{y}_\gamma^* - \mathbf{y}^k \\ \boldsymbol{\xi}_\gamma^* - \boldsymbol{\xi}^k \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \mathbf{y}_\gamma^* - \mathbf{y}^k \\ \boldsymbol{\xi}_\gamma^* - \boldsymbol{\xi}^k \end{pmatrix}^\top \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0}^\top & \gamma \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{y}_\gamma^* - \mathbf{y}^k \\ \boldsymbol{\xi}_\gamma^* - \boldsymbol{\xi}^k \end{pmatrix}. \end{aligned}$$

Note that  $g_\gamma(\boldsymbol{\theta}^k) = \mathcal{L}_\gamma(\mathbf{y}^k, \boldsymbol{\xi}^k, \boldsymbol{\theta}^k)$ , and since  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*) \in Q$ , the first-order optimality condition for  $(\mathbf{y}^k, \boldsymbol{\xi}^k)$  implies that the second term on the right-hand side of the above equality is non-negative. Therefore,

$$\begin{aligned} p_\gamma^* &\geq p_\gamma^* - \sum_{(i,j) \in \mathcal{G}} \langle \boldsymbol{\theta}_{ij}^k, A_1^{ij} \mathbf{y}_\gamma^* + A_2^{ij} \boldsymbol{\xi}_\gamma^* \rangle \\ &= \mathcal{L}_\gamma(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*, \boldsymbol{\theta}^k) \geq g_\gamma(\boldsymbol{\theta}^k) + \frac{1}{2} \left( \|\mathbf{y}_\gamma^* - \mathbf{y}^k\|_2^2 + \gamma \|\boldsymbol{\xi}_\gamma^* - \boldsymbol{\xi}^k\|_2^2 \right), \end{aligned}$$

where the first inequality above follows from  $\boldsymbol{\theta}^k \geq 0$  and  $A_1^{ij} \mathbf{y}_\gamma^* + A_2^{ij} \boldsymbol{\xi}_\gamma^* \geq 0$  for all  $(i, j) \in \mathcal{G}$  – since  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  satisfies all constraints in (7).  $\square$

**Corollary 1** *Let  $\gamma = 0$ , and  $\{\boldsymbol{\theta}^k\}$  be some dual sequence such that  $\boldsymbol{\theta}^k \geq 0$  for  $k \geq 1$  and  $\lim_{k \in \mathbb{Z}_+} g_0(\boldsymbol{\theta}^k) = p_0^*$ , i.e.,  $p_0^* = \frac{1}{2} \|\mathbf{y}^* - \bar{\mathbf{y}}\|_2^2$ , where  $\mathbf{y}^*$  is the unique optimal solution defined in (5). Moreover, let  $(\mathbf{y}^k, \boldsymbol{\xi}^k)$  denote an optimal solution to the minimization problem in (9) when  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}^k$  for  $k \geq 1$ .  $\mathbf{y}^*$  is the unique limit point of the primal sequence  $\{\mathbf{y}^k\}$ . More specifically, we have*

$$\|\mathbf{y}^k - \mathbf{y}^*\|_2^2 \leq 2(p_0^* - g_0(\boldsymbol{\theta}^k)) \rightarrow 0. \quad (18)$$



In the rest of the paper, we design methods based on dual decomposition to solve the convex regression problem in (3) or its regularized version in (7) when  $N$  is large. Suppose  $N$  is so large that solving either (3), or (7) using IPM is infeasible due to high memory requirements caused by  $\mathcal{O}(N^2)$  shape constraints. In this scenario, using dual decomposition methods, including the methods proposed in this paper, reduces the memory overhead; but, this will come at the cost of considerable increase in the run time if a high-accuracy solution is desired. That being said, in many applications, low-to-moderate-accuracy approximate solutions usually have significant value to the practitioner; this is when dual decomposition based first-order methods become attractive. Therefore, it is important to understand how the approximation quality of iterate sequence  $\{(\mathbf{y}^k, \boldsymbol{\xi}^k)\}$  changes as the algorithm runs, in order to better assess the trade of between memory requirement and convergence rate of the method chosen.

Our first objective is to study the rate of convergence in more detail. In particular, Corollary 1 implies that the projected subgradient method discussed in Section 2.2 guarantees  $\|\mathbf{y}^k - \mathbf{y}^*\|_2^2 = \mathcal{O}(1/\sqrt{k})$  rate. On the other hand, inspired by Nesterov's smoothing for solving structured non-smooth problems in [31], we can improve the convergence rate. Indeed, combining the result of Lemma 1 with Theorem 1 we see that the convergence rate in function values for the smoothed dual problem in (10) implies  $\mathbf{y}^k \rightarrow \mathbf{y}^*$ , and an  $\epsilon$ -optimal solution  $\mathbf{y}_\epsilon$ , i.e.,  $\|\mathbf{y}_\epsilon - \mathbf{y}^*\|_2^2 \leq \epsilon$ , can be computed in  $\mathcal{O}(1/\epsilon)$  iterations.

Our second objective is to study the convergence behavior of  $\{\boldsymbol{\xi}^k\}$  sequence. As discussed before in Section 2.2, when  $\gamma = 0$ , using the projected subgradient method cannot guarantee the asymptotic feasibility of  $\{(\mathbf{y}^k, \boldsymbol{\xi}^k)\}$ ; in particular, although  $y_\ell^k \rightarrow y_\ell^* = \hat{f}_N(\bar{x}_\ell)$  for all  $\ell \in \mathcal{N}$ ,  $\{\boldsymbol{\xi}_\ell^k\} \subset \mathbb{R}^n$  may not converge to a point in  $\partial \hat{f}_N(\bar{x}_\ell)$  for some  $\ell \in \mathcal{N}$ . This might be an issue to consider when designing algorithms for convex regression, as for some applications having error bounds on how  $\{\boldsymbol{\xi}_\ell^k\}$  approximates a subgradient at  $\bar{x}_\ell$  might be as important as having error bounds on how  $\{y_\ell^k\}$  approximates the function value at  $\bar{x}_\ell$ . For instance, when the objective is to fit concave utility functions to consumer data, subgradients can be used to infer consumers' marginal utilities.

These two objectives motivate the next section, where we briefly state a first-order algorithm to efficiently solve the smoothed dual problem in (9).

**Algorithm APG**( $\theta^0$ )

Iteration 0: Take  $\tilde{\theta}^1 = \theta^0$ ,  $t_1 = 1$

Iteration  $k$ : ( $k \geq 1$ ) Compute

- 1:  $\theta^k \leftarrow \Pi_{\mathcal{Q}} \left( \tilde{\theta}^k + \frac{1}{L} \nabla \rho(\tilde{\theta}^k) \right)$
- 2:  $t_{k+1} \leftarrow (1 + \sqrt{1 + 4t_k^2})/2$
- 3:  $\tilde{\theta}^{k+1} \leftarrow \theta^k + \frac{t_k - 1}{t_{k+1}} (\theta^k - \theta^{k-1})$

Fig. 1: Accelerated Proximal Gradient Algorithm

## 2.4 Parallel Accelerated Proximal Gradient (P-APG) Algorithm

Let  $\rho : \mathbb{R}^d \rightarrow \mathbb{R}$  be a concave function such that  $\nabla \rho$  is Lipschitz continuous on  $\mathbb{R}^d$  with constant  $L$ , and  $\mathcal{Q} \subset \mathbb{R}^d$  be a convex set. Given an initial iterate  $\theta^0$ , let  $\{\theta^k\}$  be the iterate sequence generated using the gradient ascent method as follows:  $\theta^{k+1} = \theta^k + \nabla \rho(\theta^k)/L$  for  $k \geq 0$ . According to Corollary 2.1.2 in [29], the error bound is given by

$$0 \leq \rho^* - \rho(\theta^k) \leq \frac{2L}{k+4} \|\theta^0 - \theta^*\|_2^2, \quad (19)$$

for all  $k \geq 1$  and for any  $\theta^* \in \arg \min\{\rho(\theta) : \theta \in \mathcal{Q}\}$ , where  $\rho^* = \rho(\theta^*)$ . On the other hand, the APG algorithm, [5, 37], displayed in Fig. 1 is based on Nesterov's accelerated gradient method [29, 31]. Corollary 3 in [37], and Theorem 4.4 in [5] show that for all  $k \geq 1$  the error bound for APG is given by

$$0 \leq \rho^* - \rho(\theta^k) \leq \frac{2L}{(k+1)^2} \|\theta^0 - \theta^*\|_2^2, \quad (20)$$

where  $\theta^0$  is the initial APG iterate and  $\theta^* \in \arg \min_{\theta \in \mathcal{Q}} \rho(\theta)$ . Hence, using APG one can compute an  $\delta$ -optimal solution within at most  $\mathcal{O}(\sqrt{L/\delta})$  APG iterations. Next, we will customize APG algorithm for solving (7) when  $\gamma > 0$ .

**Definition 3** Let  $A_3$  and  $A_4$  denote the matrices formed by vertically concatenating  $A_1^{ij}$  and  $A_2^{ij}$ , respectively, for all  $(i, j) \in \mathcal{G}$ . Define  $C \triangleq [A_3 \ A_4]$ , the decision variable vector  $\boldsymbol{\eta}^\top \triangleq [\mathbf{y}^\top \ \boldsymbol{\xi}^\top]$ , and the following elements related to the regularized problem in (7). For  $i \in \mathcal{K}$ ,  $\mathcal{Q}_i = \{(\mathbf{y}_i, \boldsymbol{\xi}_i) : \bar{A}_1^{ii} \mathbf{y}_i + \bar{A}_2^{ii} \boldsymbol{\xi}_i \geq 0\}$  and

$$\mathcal{Q} \triangleq \{\boldsymbol{\eta} = (\mathbf{y}, \boldsymbol{\xi}) : (\mathbf{y}_i, \boldsymbol{\xi}_i) \in \mathcal{Q}_i, i \in \mathcal{K}\}.$$

Now, consider the equivalent representation of (7):

$$\min_{\boldsymbol{\eta} \in \mathcal{Q}} \quad \frac{1}{2} \|\mathbf{y} - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}\|_2^2 \quad \text{s.t.} \quad C\boldsymbol{\eta} \geq 0. \quad (21)$$

The objective function in (9) for the dual problem in (10), i.e.,  $\max\{g_\gamma(\boldsymbol{\theta}) : \boldsymbol{\theta} \geq 0\}$ , can be written as

$$g_\gamma(\boldsymbol{\theta}) = \min_{\boldsymbol{\eta} \in \mathcal{Q}} \left\{ \frac{1}{2} \|\mathbf{y} - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}\|_2^2 - \langle \boldsymbol{\theta}, C\boldsymbol{\eta} \rangle \right\}. \quad (22)$$

Theorem 7.1 in [30] and Danskin's theorem imply that

$$\nabla g_\gamma(\boldsymbol{\theta}) = -C\boldsymbol{\eta}(\boldsymbol{\theta}), \quad (23)$$

where  $\boldsymbol{\eta}(\boldsymbol{\theta})$  is the unique minimizer in (22), and  $\nabla g_\gamma(\boldsymbol{\theta})$  is Lipschitz continuous with constant  $L_\gamma$  in (24), where  $\|C\|$  denotes the spectral norm of  $C$ .

$$L_\gamma = \frac{1}{\gamma} \sigma_{\max}^2(C) = \frac{1}{\gamma} \|C\|^2. \quad (24)$$

**Algorithm P-APG**( $\gamma, \theta^0$ )

Iteration 0: Set  $\tilde{\theta}^1 = \theta^0$ ,  $t_1 = 1$  and  $L_\gamma = \frac{1}{\gamma} \sigma_{\max}^2(C)$

Iteration  $k$ : ( $k \geq 1$ ) Compute

$$1: \boldsymbol{\eta}^k \leftarrow \arg \min_{\boldsymbol{\eta} \in Q} \left\{ \frac{1}{2} \|\mathbf{y} - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\boldsymbol{\xi}\|_2^2 - \langle C^\top \tilde{\boldsymbol{\theta}}^k, \boldsymbol{\eta} \rangle \right\}$$

$$2: \boldsymbol{\theta}^k \leftarrow \left( \tilde{\boldsymbol{\theta}}^k - \frac{1}{L_\gamma} C \boldsymbol{\eta}^k \right)_+$$

$$3: t_{k+1} \leftarrow (1 + \sqrt{1 + 4t_k^2})/2$$

$$4: \tilde{\boldsymbol{\theta}}^{k+1} \leftarrow \boldsymbol{\theta}^k + \frac{t_k - 1}{t_{k+1}} (\boldsymbol{\theta}^k - \boldsymbol{\theta}^{k-1})$$

Fig. 2: Parallel APG Algorithm (P-APG)

Parallel APG algorithm (P-APG), displayed in Fig. 2, is the customized version of APG algorithm in Fig. 1 to solve (10). Note that the computation in Step-1 can be carried out in *parallel* using  $K$  processors, each solving a small-size QP. Later in Section 2.5, we discuss the computational complexity of one P-APG iteration in detail.

*Adaptive Step Size Strategy:* One important property of APG methods is the ability to adopt an adaptive step-size sequence. Note  $L_\gamma$ , the Lipschitz constant of  $\nabla g_\gamma(\theta)$ , may not be known in advance or may be too conservative in practice – leading to very small steps. Instead of constant step size  $1/L_\gamma$  in Step-2 of P-APG, if one uses an adaptive step sequence  $\{1/s_k\}$ , the  $\mathcal{O}(L_\gamma/k^2)$  rate shown in [5] still holds as long as

$$g_\gamma(\boldsymbol{\theta}^k) \geq g_\gamma(\tilde{\boldsymbol{\theta}}^k) + \langle \nabla g_\gamma(\tilde{\boldsymbol{\theta}}^k), \boldsymbol{\theta}^k - \tilde{\boldsymbol{\theta}}^k \rangle - \frac{s_k}{2} \|\boldsymbol{\theta}^k - \tilde{\boldsymbol{\theta}}^k\|_2^2, \quad (25)$$

holds for all  $k$  where  $\boldsymbol{\theta}^k$  is computed using  $s_k$  instead of  $1/L_\gamma$ . Clearly, one can choose  $s^k \leq L_\gamma$ ; possibly take longer steps compared to constant step size  $1/L_\gamma$  and still has a convergence guarantee with the same rate. We adopted the following rule in our numerical tests: let  $v > 1$ , for  $k \geq 1$  we set  $s_k = s_{k-1} v^{\ell_k - 1}$  where  $\ell_k \geq 0$  is the smallest integer such that (25) holds, and  $s_0 = L_\gamma$ .

In the rest of the paper, other than the numerical section, for the sake of simplicity we assume  $s_k = L_\gamma$  for all  $k$ . To better understand the convergence rate of P-APG, next, we provide a bound on  $\|\boldsymbol{\theta}_\gamma^*\|_2$  for all  $\boldsymbol{\theta}_\gamma^* \in \Theta_\gamma^*$ .

**Lemma 2** *Given  $\gamma \geq 0$  and  $\delta \geq 0$ , let  $\boldsymbol{\theta}_{\gamma,\delta} \geq 0$  be a  $\delta$ -optimal solution to (10), i.e.,  $0 \leq p_\gamma^* - g_\gamma(\boldsymbol{\theta}_{\gamma,\delta}) \leq \delta$ . Given  $\{(\bar{x}_\ell, \bar{y}_\ell)\}_{\ell \in \mathcal{N}}$ , for all  $\ell \in \mathcal{N}$ , define  $\tilde{y}_\ell \triangleq \hat{y} + \frac{\alpha}{2} \|\bar{x}_\ell - \hat{x}\|_2^2$  for some given  $\alpha > 0$ , where  $\hat{x} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \bar{x}_\ell$  and  $\hat{y} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \bar{y}_\ell$ . Then*

$$\|\boldsymbol{\theta}_{\gamma,\delta}\|_1 \leq \frac{2}{\alpha v} \left( \delta - p_\gamma^* + \frac{1}{2} \sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \bar{y}_\ell)^2 + \gamma \alpha^2 \|\bar{x}_\ell - \hat{x}\|_2^2 \right) \triangleq B(\gamma, \delta, \alpha), \quad (26)$$

where  $v \triangleq \min_{(i,j) \in \mathcal{G}} \{\|\bar{x}_{\ell_1} - \bar{x}_{\ell_2}\|_2^2 : \ell_1 \in \mathcal{C}_i, \ell_2 \in \mathcal{C}_j\}$ .

*Proof* For given  $\alpha > 0$ , define  $h : \mathbb{R}^n \rightarrow \mathbb{R}$  such that  $h(x) \triangleq \hat{y} + \frac{\alpha}{2} \|x - \hat{x}\|_2^2$ . Note that for all  $\ell \in \mathcal{N}$ , we have  $\tilde{y}_\ell = h(\bar{x}_\ell)$ , and  $\tilde{\xi}_\ell \triangleq \nabla h(\bar{x}_\ell) = \alpha(\bar{x}_\ell - \hat{x})$ . Since  $h$  is strongly convex with modulus  $\alpha > 0$ , for any  $(\ell_1, \ell_2) \in \mathcal{N} \times \mathcal{N}$ , it follows that

$$\tilde{y}_{\ell_2} - \tilde{y}_{\ell_1} + \left\langle \tilde{\xi}_{\ell_1}, \bar{x}_{\ell_1} - \bar{x}_{\ell_2} \right\rangle \geq \frac{\alpha}{2} \|\bar{x}_{\ell_2} - \bar{x}_{\ell_1}\|_2^2 \geq 0. \quad (27)$$

Let  $\tilde{\boldsymbol{\eta}} = [\tilde{\mathbf{y}}^\top \tilde{\boldsymbol{\xi}}^\top]^\top$  such that  $\tilde{\mathbf{y}} = [\tilde{y}_i]_{i \in \mathcal{K}}$  and  $\tilde{\boldsymbol{\xi}} = [\tilde{\xi}_i]_{i \in \mathcal{K}}$ , where  $\tilde{y}_i = [\tilde{y}_\ell]_{\ell \in \mathcal{C}_i}$  and  $\tilde{\xi}_i = [\tilde{\xi}_\ell]_{\ell \in \mathcal{C}_i}$ . Hence,  $\tilde{\boldsymbol{\eta}} \in Q$  is a Slater point for the problem in (7), or equivalently (21). Since  $C\tilde{\boldsymbol{\eta}} \geq \frac{\alpha v}{2} \mathbf{1} > 0$ , it follows from (22) that

$$\frac{\alpha v}{2} \|\boldsymbol{\theta}_{\gamma, \delta}\|_1 \leq \langle \boldsymbol{\theta}_{\gamma, \delta}, C\tilde{\boldsymbol{\eta}} \rangle \leq \frac{1}{2} \|\tilde{\mathbf{y}} - \bar{\mathbf{y}}\|_2^2 + \frac{\gamma}{2} \|\tilde{\boldsymbol{\xi}}\|_2^2 - g_\gamma(\boldsymbol{\theta}_{\gamma, \delta}), \quad (28)$$

and the result follows from  $\delta$ -optimality, i.e.,  $p_\gamma^* - g_\gamma(\boldsymbol{\theta}_{\gamma, \delta}) \leq \delta$ .  $\square$

*Remark 1* When  $\gamma = 0$ , for any  $\boldsymbol{\theta}_0^* \in \Theta_0^*$ , it follows that

$$\|\boldsymbol{\theta}_0^*\|_1 \leq B(0, 0, \alpha) = \frac{2}{\alpha v} \left( \frac{1}{2} \sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \bar{y}_\ell)^2 - p_0^* \right) \leq \frac{1}{\alpha v} \sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \bar{y}_\ell)^2 \triangleq B_\theta(\alpha).$$

Note  $p_0^* \leq p_\gamma^*$  for all  $\gamma \geq 0$ ; hence, when  $\gamma > 0$ , for any  $\boldsymbol{\theta}_\gamma^* \in \Theta_\gamma^*$ , it follows that

$$\begin{aligned} \|\boldsymbol{\theta}_\gamma^*\|_1 &\leq B(\gamma, 0, \alpha) \leq B(0, 0, \alpha) + \frac{\gamma \alpha}{v} \sum_{\ell \in \mathcal{N}} \|\bar{x}_\ell - \hat{x}\|_2^2 \\ &\leq B_\theta(\alpha) + \frac{\gamma \alpha}{v} \sum_{\ell \in \mathcal{N}} \|\bar{x}_\ell - \hat{x}\|_2^2 \triangleq B_\theta(\gamma, \alpha). \end{aligned}$$

The bound on  $\|\boldsymbol{\theta}_{\gamma, \delta}\|_1$  given in (26) holds for all  $\alpha > 0$ . Therefore, by choosing  $\alpha > 0$  depending on  $\gamma \geq 0$ , we optimize the upper bounds  $B_\theta(\alpha)$  and  $B_\theta(\gamma, \alpha)$  defined in Remark 1.

**Lemma 3** Given  $\gamma \geq 0$ , let  $\alpha_\gamma^* \triangleq \arg \min\{B_\theta(\gamma, \alpha) : \alpha > 0\}$ , and  $\alpha^* \triangleq \alpha_0^*$  for  $\gamma = 0$ , i.e.,  $\alpha^* = \arg \min\{B_\theta(\alpha) : \alpha > 0\}$ . For any  $\gamma \geq 0$ ,  $\alpha_\gamma^*$  can be computed as follows

$$\alpha_\gamma^* = 4 \left( \frac{\sum_{\ell \in \mathcal{N}} (\tilde{y}_\ell - \hat{y})^2}{\sum_{\ell \in \mathcal{N}} \|\bar{x}_\ell - \hat{x}\|_2^2 (\|\bar{x}_\ell - \hat{x}\|_2^2 + 8\gamma)} \right)^{1/2}, \quad (29)$$

leading to tight upper bounds  $B_\theta^* \triangleq B_\theta(\alpha^*)$  and  $B_\theta^*(\gamma) \triangleq B_\theta(\gamma, \alpha_\gamma^*)$ .

*Proof* According to definition of  $\{\tilde{y}_\ell\}_{\ell \in \mathcal{N}}$  given in Lemma 2,  $B_\theta(\gamma, \alpha)$  can be explicitly stated as follows:

$$B_\theta(\gamma, \alpha) = \frac{1}{\alpha v} \sum_{\ell \in \mathcal{N}} \left( \hat{y} - \bar{y}_\ell + \frac{\alpha}{2} \|\hat{x} - \bar{x}_\ell\|_2^2 \right)^2 + \frac{\gamma \alpha}{v} \sum_{\ell \in \mathcal{N}} \|\hat{x} - \bar{x}_\ell\|_2^2. \quad (30)$$

To simplify the notation, let  $p_\ell \triangleq \hat{y} - \bar{y}_\ell$ , and  $q_\ell \triangleq \frac{1}{2} \|\hat{x} - \bar{x}_\ell\|_2^2$  for  $\ell \in \mathcal{N}$ . Via the change of variables  $\beta = \sqrt{\alpha}$ , we obtain the following equivalent problem:

$$\min_{\beta} \left\{ w(\beta) \triangleq \sum_{\ell \in \mathcal{N}} \left( \frac{1}{\beta} p_\ell + \beta q_\ell \right)^2 + 2\gamma\beta^2 \sum_{\ell \in \mathcal{N}} q_\ell : \beta > 0 \right\}. \quad (31)$$

Clearly, we have

$$w'(\beta) = \sum_{\ell \in \mathcal{N}} q_\ell (q_\ell + 4\gamma) \beta - \frac{p_\ell^2}{\beta^3}, \quad w''(\beta) = \sum_{\ell \in \mathcal{N}} q_\ell (q_\ell + 4\gamma) + 3 \frac{p_\ell^2}{\beta^4}. \quad (32)$$

Since  $w''(\beta) \geq 0$  for  $\beta > 0$ ,  $w(\beta)$  is a convex function and first-order necessary optimality condition, i.e.,  $w'(\beta^*) = 0$ , is also sufficient. In particular, solving for  $\beta^*$  and setting  $\alpha_\gamma^* = \sqrt{\beta^*}$  gives the desired result in (29).  $\square$

Let constants  $B_\theta^*$  and  $B_\theta^*(\gamma)$  be as defined in Lemma 3. Now, using (24) and the bounds given in Remark 1, we can customize the generic rate results in (19) for *gradient ascent* and those in (20) for APG methods. In particular, for any  $\gamma > 0$ , in order to compute a  $\delta$ -optimal solution to the problem in (10), i.e.,  $\theta_{\gamma,\delta} \geq 0$  such that  $0 \leq p_\gamma^* - g_\gamma(\theta_{\gamma,\delta}) \leq \delta$ , the gradient ascent method requires  $\mathcal{O}(L_\gamma/\delta) = \mathcal{O}(B_\theta^{*2}/(\gamma\delta))$  iterations. On the other hand, P-APG in Fig. 2 can compute a  $\delta$ -optimal solution to (10) within  $\mathcal{O}(\sqrt{L_\gamma/\delta})$  iterations. More precisely, (24) implies  $\mathcal{O}(B_\theta^*(\gamma)/(\gamma\delta)^{1/2})$  iteration complexity for P-APG when applied to (10).

The  $\mathcal{O}(1)$  constant depends on  $\sigma_{\max}(C)$ , and to better have a better understanding of how it grows with the problem size, we provide some bounds for  $\sigma_{\max}(A_1)$ ,  $\sigma_{\max}(A_2)$ , and  $\sigma_{\max}(C)$ .

**Lemma 4** *Let  $A_1 \in \mathbb{R}^{N(N-1) \times N}$  and  $A_2 \in \mathbb{R}^{N(N-1) \times Nn}$  be the matrices in (4), i.e., corresponding to the constraints in (3); and let  $A_3$ ,  $A_4$ , and  $C$  be the matrices as given in Definition 3. Then,  $\sigma_{\max}(A_3) \leq \sigma_{\max}(A_1) = \sqrt{2N}$ ,  $\sigma_{\max}(A_4) \leq \sigma_{\max}(A_2) \leq B_x N$ , and  $\sigma_{\max}(C) \leq \sqrt{2N} + B_x N$ .*

*Proof* It is easy to observe that  $A_1^\top A_1 = 2\Omega$ , where  $\Omega \in \mathbb{R}^{N \times N}$  denotes the Laplacian matrix of a complete graph with  $N$  vertices, i.e., for each  $i = 1, \dots, N$ ,  $\Omega_{ii} = N - 1$ , and  $\Omega_{ij} = -1$  for all  $j \neq i$ . It is known that  $\Omega$  has two distinct eigenvalues: 0 (with multiplicity 1) and  $N$  (with multiplicity  $N - 1$ ). Therefore,  $\sigma_{\max}(A_1) = \sqrt{2N}$ ; and since  $A_3$  is a submatrix of  $A_1$ , one immediately has  $\sigma_{\max}(A_3) \leq \sigma_{\max}(A_1)$ .

Since  $A_2$  is block-diagonal, we have  $\sigma_{\max}(A_2) = \max_{\ell \in \mathcal{N}} \{\sigma_{\max}(X_\ell)\}$ , where  $X_\ell = -T_\ell \bar{X}$  (see Definition 1 for  $T_\ell$  and  $\bar{X}$ ). Hence, we have  $\sigma_{\max}(A_2) \leq \|\bar{X}\| \max_{\ell \in \mathcal{N}} \{\|T_\ell\|\}$ . For  $\ell \in \mathcal{N}$ , let  $\Omega_\ell = T_\ell^\top T_\ell$ ; it is easy to observe that  $\Omega_\ell \in \mathbb{R}^{N \times N}$  is the Laplacian matrix of a star-tree with  $N - 1$  leaves ( $\ell$  is the internal node). It is known that  $\Omega$  has three distinct eigenvalues: 0 (with multiplicity 1), 1 (with multiplicity  $N - 2$ ), and  $N$  (with multiplicity 1). Therefore,  $\|T_\ell\| = \sqrt{N}$  for all  $\ell \in \mathcal{N}$ . On the other hand, since  $\|\bar{x}_\ell\|_2 \leq B_x$  for  $\ell \in \mathcal{N}$ ,  $\|\bar{X}\| \leq B_x \sqrt{N}$ . Therefore,  $\sigma_{\max}(A_2) \leq B_x N$ ; and since  $A_4$  is a submatrix of  $A_2$ , one immediately has  $\sigma_{\max}(A_4) \leq \sigma_{\max}(A_2)$ . Finally, since  $C = [A_3 A_4]$ , clearly  $\|C\| \leq \|A_3\| + \|A_4\|$ .  $\square$

Next, we study the error bounds for inexact solutions. Given  $\gamma > 0$ , let  $\boldsymbol{\theta}_{\gamma,\delta}$  be a  $\delta$ -optimal solution to (10), and  $(\mathbf{y}_{\gamma,\delta}, \boldsymbol{\xi}_{\gamma,\delta})$  be the optimal solution to the minimization problem in (9), or equivalently to (22), when  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}_{\gamma,\delta}$ . In Theorem 2 we establish error bounds on the suboptimality  $\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2$ , and on the infeasibility  $\|(A_1 \mathbf{y}_{\gamma,\delta} + A_2 \boldsymbol{\xi}_{\gamma,\delta})_-\|_2$ .

**Theorem 2** *Given  $\gamma > 0$ , let  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  and  $\boldsymbol{\theta}_\gamma^*$  denote the optimal solutions to (7) and (10), respectively. Let  $\boldsymbol{\theta}_{\gamma,\delta}$  be a  $\delta$ -optimal solution to (10), and  $(\mathbf{y}_{\gamma,\delta}, \boldsymbol{\xi}_{\gamma,\delta})$  be the minimizer in (22) when  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}_{\gamma,\delta}$ . For all  $\gamma, \delta > 0$ , the following bounds hold:*

$$\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2 \leq \|\boldsymbol{\xi}^*\|_2 \sqrt{\gamma} + \sqrt{2\delta}, \quad (33)$$

$$\|(A_1 \mathbf{y}_{\gamma,\delta} + A_2 \boldsymbol{\xi}_{\gamma,\delta})_-\|_2 \leq 2\sqrt{N\delta} + B_x N \sqrt{\frac{2\delta}{\gamma}}. \quad (34)$$

Moreover, both starting from the initial iterate  $\boldsymbol{\theta}^0 = \mathbf{0}$ , P-APG can compute  $(\mathbf{y}_{\gamma,\delta}, \boldsymbol{\xi}_{\gamma,\delta})$  within  $K(\delta, \gamma) = \sigma_{\max}(C) B_\theta^*(\gamma) \sqrt{2/(\gamma\delta)}$  iterations while gradient ascent requires  $2(B_\theta^* \sigma_{\max}(C))^2/(\gamma\delta)$  iterations, where  $\sigma_{\max}(C) = \mathcal{O}(N)$ .

*Proof* Given  $\boldsymbol{\theta}_{\gamma,\delta} \geq 0$  and the corresponding minimizer,  $(\mathbf{y}_{\gamma,\delta}, \boldsymbol{\xi}_{\gamma,\delta})$ , to the problem in (22) when  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}_{\gamma,\delta}$ , Theorem 1 implies that

$$\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}_\gamma^*\|_2^2 + \gamma \|\boldsymbol{\xi}_{\gamma,\delta} - \boldsymbol{\xi}_\gamma^*\|_2^2 \leq 2(p_\gamma^* - g_\gamma(\boldsymbol{\theta}_{\gamma,\delta})) \leq 2\delta. \quad (35)$$

Hence, Lemma 1 and (35) together imply that

$$\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2 \leq \|\mathbf{y}_{\gamma,\delta} - \mathbf{y}_\gamma^*\|_2 + \|\mathbf{y}_\gamma^* - \mathbf{y}^*\|_2 \leq \|\boldsymbol{\xi}^*\|_2 \sqrt{\gamma} + \sqrt{2\delta}.$$

Moreover, since  $(\mathbf{y}_{\gamma,\delta}, \boldsymbol{\xi}_{\gamma,\delta}) \in Q$  and  $(\mathbf{y}_\gamma^*, \boldsymbol{\xi}_\gamma^*)$  is feasible to (7), i.e.,  $A_1 \mathbf{y}_\gamma^* + A_2 \boldsymbol{\xi}_\gamma^* \geq 0$ , we have

$$\begin{aligned} & \|(A_1 \mathbf{y}_{\gamma,\delta} + A_2 \boldsymbol{\xi}_{\gamma,\delta})_-\|_2 \\ &= \|(A_3 \mathbf{y}_{\gamma,\delta} + A_4 \boldsymbol{\xi}_{\gamma,\delta})_-\|_2 = \|(A_3 \mathbf{y}_{\gamma,\delta} + A_4 \boldsymbol{\xi}_{\gamma,\delta}) - (A_3 \mathbf{y}_\gamma^* + A_4 \boldsymbol{\xi}_\gamma^*)_-\|_2, \\ &\leq \|A_3 (\mathbf{y}_{\gamma,\delta} - \mathbf{y}_\gamma^*) + A_4 (\boldsymbol{\xi}_{\gamma,\delta} - \boldsymbol{\xi}_\gamma^*)\|_2 \\ &\leq \sigma_{\max}(A_3) \|\mathbf{y}_{\gamma,\delta} - \mathbf{y}_\gamma^*\|_2 + \sigma_{\max}(A_4) \|\boldsymbol{\xi}_{\gamma,\delta} - \boldsymbol{\xi}_\gamma^*\|_2 \end{aligned} \quad (36)$$

where the first inequality follows from the fact that  $\|\mathbf{x} - \mathbf{y}\|_2 \geq \|(\mathbf{x})_- - (\mathbf{y})_-\|_2$  for any  $\mathbf{x}$  and  $\mathbf{y}$ . The infeasibility result in (34) immediately follows from (35) and (36). The iteration complexity bounds can be obtained using the arguments immediately after Remark 1.  $\square$

As for some applications having an error bound on how  $\boldsymbol{\xi}_{\gamma,\delta}$  approximates  $\boldsymbol{\xi}^*$ , i.e., the subgradients at  $\{\bar{x}_\ell\}_{\ell \in \mathcal{N}}$ , is crucial. Next, we show that  $\|\boldsymbol{\xi}_{\gamma,\delta} - \boldsymbol{\xi}^*\|_2$  is indeed small.

**Theorem 3** *There exists  $K > 0$  such that  $\|\xi_\gamma^* - \xi^*\|_2 \leq K \|A_1(\mathbf{y}_\gamma^* - \mathbf{y}^*)\|_2$ ; hence,  $\|\xi_\gamma^* - \xi^*\|_2 \leq K \sigma_{\max}(A_1) \|\xi^*\|_2 \sqrt{\gamma}$ , which implies*

$$\|\xi_{\gamma,\delta} - \xi^*\|_2 \leq K \|\xi^*\|_2 \sqrt{2N\gamma} + \sqrt{\frac{2\delta}{\gamma}}.$$

*Proof* Since  $\mathbf{y}^*$  is the unique optimal solution to (4), (5) implies that  $\xi^* = \arg \min\{\|\xi\|_2 : A_1 \mathbf{y}^* + A_2 \xi \geq \mathbf{0}\}$ . Similarly, (7) implies that  $\xi_\gamma^* = \arg \min\{\|\xi\|_2 : A_1 \mathbf{y}_\gamma^* + A_2 \xi \geq \mathbf{0}\}$ . Define  $\mathbf{h}(\gamma) \triangleq -A_1 \mathbf{y}_\gamma^*$  for  $\gamma \geq 0$ . Note from Lemma 1, we have  $\mathbf{h}(0) = -A_1 \mathbf{y}_0^* = -A_1 \mathbf{y}^*$ . Therefore, for  $\gamma \geq 0$ ,

$$\xi_\gamma^* = \arg \min \{ \|\xi\|_2 : A_2 \xi \geq \mathbf{h}(\gamma) \}. \quad (37)$$

Note that for  $\gamma = 0$ ,  $\xi_0^* = \xi^*$ . Sensitivity of metric projection onto parametric polyhedral sets is studied in [38]. According to Theorem 2.1 in [38], there exists  $K > 0$  such that

$$\|\xi_\gamma^* - \xi_{\gamma'}^*\|_2 \leq K \|\mathbf{h}(\gamma) - \mathbf{h}(\gamma')\|_2 \leq K \sigma_{\max}(A_1) \|\mathbf{y}_\gamma^* - \mathbf{y}_{\gamma'}^*\|_2, \quad \forall \gamma, \gamma' \geq 0. \quad (38)$$

Therefore, given  $\gamma > 0$ , setting  $\gamma' = 0$ , and using Lemma 1, we have

$$\|\xi_\gamma^* - \xi^*\|_2 \leq K \sigma_{\max}(A_1) \|\mathbf{y}_\gamma^* - \mathbf{y}^*\|_2 \leq K \sigma_{\max}(A_1) \|\xi^*\|_2 \sqrt{\gamma}. \quad (39)$$

Moreover, (35) implies that  $\|\xi_{\gamma,\delta} - \xi_\gamma^*\|_2 \leq \sqrt{\frac{2\delta}{\gamma}}$ . Hence, combining this with (39) gives the desired result since  $\sigma_{\max}(A_1) = \sqrt{2N}$ .  $\square$

We can summarize Theorem 2 and Theorem 3 briefly as follows. If the main objective is the function value approximation and estimating the subgradients are not crucial, then according to Theorem 2, for any given  $\epsilon > 0$ , setting  $\gamma = \delta = \epsilon$  implies that  $\mathbf{y}_{\gamma,\delta} \in \mathbb{R}^N$  satisfies  $\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2^2 = \mathcal{O}(\epsilon)$  and it can be computed within  $\mathcal{O}(N^2 B_\theta^{*2}/\epsilon^2)$  iterations of the gradient ascent method on (10) (which is the *same* as the iteration complexity of the projected subgradient method applied to (10) for  $\gamma = 0$ ), and within  $\mathcal{O}(NB_\theta^*(\epsilon)/\epsilon)$  iterations of P-APG in Fig. 2 on (10). On the other hand if the subgradient approximation is important too, then according to Lemma 1, Theorem 2 and Theorem 3, for any given  $\epsilon > 0$ , by setting  $\gamma = \epsilon$  and  $\delta = \epsilon^2$  implies that  $\mathbf{y}_{\gamma,\delta} \in \mathbb{R}^N$  satisfies  $\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2^2 = \mathcal{O}(\epsilon)$ ,  $\|\xi_{\gamma,\delta} - \xi^*\|_2^2 = \mathcal{O}(\epsilon)$  and  $\|(A_1 \mathbf{y}_{\gamma,\delta} + A_2 \xi_{\gamma,\delta}) - \mathbf{0}\|_2^2 \leq \mathcal{O}(\epsilon)$  within  $\mathcal{O}(N^2 B_\theta^{*2}/\epsilon^3)$  iterations using the gradient ascent method on (10), and within  $\mathcal{O}(NB_\theta^*(\epsilon)/\epsilon^{3/2})$  iterations using P-APG in Fig. 2 on (10).

#### 2.4.1 Continuation Method for Convex Regression

Let  $\theta_{\gamma,\delta}$  be a  $\delta$ -optimal solution to (10), and  $(\mathbf{y}_{\gamma,\delta}, \xi_{\gamma,\delta})$  be the minimizer in (22) when  $\theta$  is set to  $\theta_{\gamma,\delta}$ . In Section 2.4, we have seen that for any fixed  $\epsilon$ , setting  $\gamma = \delta = \epsilon$  implies that  $\mathbf{y}_{\gamma,\delta}$  can be computed within  $\mathcal{O}(NB_\theta^*(\epsilon)/\epsilon)$  iterations of P-APG and it satisfies  $\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2^2 = \mathcal{O}(\epsilon)$ . In this section, we

describe a continuation method to solve (4). In particular, we would like to generate an iterate sequence  $\{\mathbf{y}^t\}_{t \in \mathbb{Z}_+}$  such that  $\mathbf{y}^{(t)} \rightarrow \mathbf{y}^*$  as  $t \rightarrow +\infty$  with the following properties:

- i) for any  $\epsilon > 0$ ,  $\mathbf{y}^{(t)}$  satisfies  $\|\mathbf{y}^{(t)} - \mathbf{y}^*\|_2^2 = \mathcal{O}(\epsilon)$  for all  $t \geq T_\epsilon = \mathcal{O}(\log(1/\epsilon))$ ;
- ii) moreover,  $T_\epsilon$  iterations of the continuation require at most  $\mathcal{O}(1/\epsilon)$  P-APG iterations in total, i.e., the algorithm generates an asymptotically optimal iterate sequence with  $\mathcal{O}(1/\epsilon)$  rate without fixing the algorithmic parameters depending on the tolerance  $\epsilon > 0$ .

Let  $\beta > 1$  and define  $\{\epsilon_t\}_{t \in \mathbb{Z}_+}$  such that  $\epsilon_t = \epsilon_0/\beta^t$  for some  $\epsilon_0 > 0$ . Also define  $\{\gamma_t\}_{t \in \mathbb{Z}_+}$  and  $\{\delta_t\}_{t \in \mathbb{Z}_+}$  such that  $\gamma_t = \kappa_\gamma \epsilon_t$  and  $\delta_t = \kappa_\delta \epsilon_t$  for  $t \geq 1$  for some  $\kappa_\gamma, \kappa_\delta > 0$ . Next, for all  $t \geq 1$ , let  $\boldsymbol{\theta}^{(t)} \triangleq \boldsymbol{\theta}_{\gamma_t, \delta_t}$  be a  $\delta_t$ -optimal solution to (10) when  $\gamma = \gamma_t$ , such that it is computed using P-APG in Fig. 2 starting from the initial iterate  $\boldsymbol{\theta}^{(t-1)}$ , where  $\boldsymbol{\theta}^{(0)} = \mathbf{0}$ , and  $(\mathbf{y}^{(t)}, \boldsymbol{\xi}^{(t)})$  be the minimizer in (22) when  $\boldsymbol{\theta}$  is set to  $\boldsymbol{\theta}^{(t)}$  and  $\gamma = \gamma_t$ . Then clearly from (33), we have

$$\|\mathbf{y}^{(t)} - \mathbf{y}^*\|_2 \leq \|\boldsymbol{\xi}^*\|_2 \sqrt{\gamma_t} + \sqrt{2\delta_t} = \Gamma \beta^{-\frac{t}{2}}, \quad t \geq 1, \quad (40)$$

where  $\Gamma \triangleq \sqrt{\epsilon_0} (\|\boldsymbol{\xi}^*\|_2 \sqrt{\kappa_\gamma} + \sqrt{2\kappa_\delta})$ . Therefore,  $\|\mathbf{y}^{(t)} - \mathbf{y}^*\|_2^2 \leq \epsilon$  for all  $t \geq T_\epsilon \triangleq \lceil \log_\beta(\Gamma^2/\epsilon) \rceil$ . Let  $\mathbf{y}_\epsilon \triangleq \mathbf{y}^{(T_\epsilon)}$ ; hence,  $\|\mathbf{y}_\epsilon - \mathbf{y}^*\|_2^2 \leq \epsilon$ .

Note that for all  $t \geq 1$ , starting from  $\boldsymbol{\theta}^{(t-1)}$ , P-APG can compute  $\boldsymbol{\theta}^{(t)}$  within  $K_t \triangleq \sigma_{\max}(C) \|\boldsymbol{\theta}_{\gamma_t}^* - \boldsymbol{\theta}^{(t-1)}\|_2 \sqrt{2/(\delta_t \gamma_t)}$  iterations. From Lemma 2 and Remark 1, it follows that  $\|\boldsymbol{\theta}^{(t-1)}\|_1 \leq B(\gamma_{t-1}, \delta_{t-1}, \alpha) \leq B_\theta(\gamma_{t-1}, \alpha) + \frac{2\delta_{t-1}}{\alpha v}$ , and  $\|\boldsymbol{\theta}_{\gamma_t}^*\|_1 \leq B_\theta(\gamma_t, \alpha)$  for all  $\alpha > 0$ . Therefore, Lemma 3 and  $\gamma_{t-1} > \gamma_t$  imply

$$\|\boldsymbol{\theta}_{\gamma_t}^* - \boldsymbol{\theta}^{(t-1)}\|_2 \leq \|\boldsymbol{\theta}^{(t-1)}\|_1 + \|\boldsymbol{\theta}_{\gamma_t}^*\|_1 \leq 2B_\theta^*(\gamma_{t-1}) + \frac{2\delta_{t-1}}{\alpha_{\gamma_{t-1}}^* v}. \quad (41)$$

Hence,  $K_t$ , the number of P-APG iterations to compute  $\boldsymbol{\theta}^{(t)}$  can be bounded above as follows

$$K_t \leq \bar{K}_t \triangleq \|C\| \left( 2B_\theta^*(\gamma_{t-1}) + \frac{2\delta_{t-1}}{\alpha_{\gamma_{t-1}}^* v} \right) \sqrt{\frac{2}{\kappa_\gamma \kappa_\delta}} \frac{1}{\epsilon_t}. \quad (42)$$

From Lemma 3, we have  $\alpha_{\gamma_0}^* \leq \alpha_{\gamma_{t-1}}^* \leq \alpha_{\gamma_t}^* \leq \alpha^*$  for  $t \geq 1$ ; hence, for  $\gamma > 0$ ,

$$B_\theta^*(\gamma) = B_\theta(\alpha_\gamma^*) + \frac{\gamma \alpha_\gamma^*}{v} \sum_{\ell \in \mathcal{N}} \|\bar{x}_\ell - \hat{x}\|_2^2 \leq B_\theta(\alpha_{\gamma_0}^*) + \frac{\gamma \alpha_\gamma^*}{v} \sum_{\ell \in \mathcal{N}} \|\bar{x}_\ell - \hat{x}\|_2^2.$$

Using this upper bound in (42), we can bound the total number of P-APG iterations needed to compute  $\mathbf{y}_\epsilon$ . In particular,  $\mathbf{y}_\epsilon$  can be computed within

$$\sum_{t=1}^{T_\epsilon} \bar{K}_t \leq \|C\| \sqrt{\frac{2}{\kappa_\gamma \kappa_\delta}} \left[ \frac{2\beta}{v} \left( \kappa_\gamma \alpha^* \sum_{\ell \in \mathcal{N}} \|\bar{x}_\ell - \hat{x}\|_2^2 + \frac{\kappa_\delta}{\alpha_{\gamma_0}^*} \right) T_\epsilon + \frac{2B_\theta(\alpha_{\gamma_0}^*)}{\epsilon_0} \sum_{t=1}^{T_\epsilon} \beta^t \right] \quad (43)$$

P-APG iterations. Note that  $\sum_{t=1}^{T_\epsilon} \beta^k = \frac{1}{\beta-1} (\beta^{T_\epsilon} - 1) \leq \frac{\beta}{\beta-1} \frac{\Gamma^2}{\epsilon}$  since  $T_\epsilon = \lceil \log_\beta(\Gamma^2/\epsilon) \rceil$ . Therefore,  $\sum_{t=1}^{T_\epsilon} \bar{K}_t = \mathcal{O}(1/\epsilon)$ .



To implement this scheme, for each outer iteration  $t \geq 1$ , *verifiable* sufficient conditions for  $\delta_t$ -optimality can be used to terminate inner P-APG iterations. In fact, the number of P-APG iterations to compute  $\boldsymbol{\theta}^{(t)}$  is bounded above by  $\bar{K}_t$ , which can be computed a-priori; hence, giving us a stopping condition for the inner iterations. Moreover, one can also use other stopping conditions for inner iterations based on  $\nabla g_{\gamma_t}$  which are also sufficient for  $\delta_t$  optimality; thus, making it possible to proceed to the next outer iteration before waiting for  $\bar{K}_t$  inner iterations – see Section 3.3 in [3] for a similar discussion.

## 2.5 Computational complexity of P-APG iterations

In Section 2.4, we have seen that for any fixed  $\epsilon$ , setting  $\gamma = \delta = \epsilon$  implies that  $\|\mathbf{y}_{\gamma,\delta} - \mathbf{y}^*\|_2^2 = \mathcal{O}(\epsilon)$  and  $\mathbf{y}_{\gamma,\delta}$  can be computed within  $\mathcal{O}(1/\epsilon)$  iterations of P-APG – see Theorem 2. In Section 2.4.1, we discussed that using continuation one can generate an iterate sequence  $\{\mathbf{y}^t\}_{t \in \mathbb{Z}_+}$  such that  $\mathbf{y}^{(t)} \rightarrow \mathbf{y}^*$  as  $t \rightarrow +\infty$ , and  $\|\mathbf{y}^{(t)} - \mathbf{y}^*\|_2^2 = \mathcal{O}(\epsilon)$  for all  $t \geq T_\epsilon = \mathcal{O}(1/\epsilon)$  for all  $\epsilon > 0$ ; moreover, computing  $\mathbf{y}^{(T_\epsilon)}$  require at most  $\mathcal{O}(1/\epsilon)$  iterations of P-APG in total – see (43).

The bottleneck operations at each P-APG iteration, displayed in Fig. 2, are i) evaluating the matrix-vector multiplications with  $C$  and  $C^\top$ , and ii) computing Step 1, which requires solving  $K$  *small*-size QPs. Matrix-vector multiplications with  $C$  and  $C^\top$  requires evaluating multiplications with  $A_3$ ,  $A_3^\top$ ,  $A_4$  and  $A_4^\top$ . Moreover, since  $A_3$  is a submatrix of  $A_1 \in \mathbb{R}^{N(N-1) \times N}$ , and  $A_4$  is a submatrix of  $A_2 \in \mathbb{R}^{N(N-1) \times Nn}$ , as long as left and right vector multiplications with  $A_1$  and  $A_2$  can be done efficiently, one can do same operations with  $C$  easily. Due to specific structures of  $A_1$  and  $A_2$ , without forming  $A_1$  and  $A_2$  explicitly, one can compute  $A_1 \mathbf{y}$  and  $A_1^\top \mathbf{z}$  with  $\mathcal{O}(N^2 - N)$  complexity for all  $\mathbf{y}$  and  $\mathbf{z}$ ;  $A_2 \boldsymbol{\xi}$  and  $A_2^\top \boldsymbol{\omega}$  with  $\mathcal{O}(n(N^2 - N))$  complexity for all  $\boldsymbol{\xi}$  and  $\boldsymbol{\omega}$ . More importantly, neither  $A_1$  nor  $A_2$  is stored in the memory; storing only  $\{\bar{x}_\ell\}_{\ell=1}^N$  is sufficient to be able to compute these matrix-vector multiplications.

First, we will consider the bottleneck step while solving (7) using a primal-dual IPM alone, without P-APG. This result will also help us understand the complexity of computing Step 1, which requires solving  $K$  small size QPs as shown in (13), which are in a similar form with the QP in (7).

Let  $c \in \mathbb{R}^{N(n+1)}$  be an arbitrary vector,  $G = \begin{bmatrix} \mathbf{I}_N & \mathbf{0} \\ \mathbf{0}^\top & \gamma \mathbf{I}_{Nn} \end{bmatrix}$ , and  $A = [A_1 \ A_2]$  where  $A_1 \in \mathbb{R}^{N(N-1) \times N}$  and  $A_2 \in \mathbb{R}^{N(N-1) \times Nn}$  are defined in (6). Consider the generic QP

$$\min_{\boldsymbol{\eta}} \frac{1}{2} \boldsymbol{\eta}^\top G \boldsymbol{\eta} + c^\top \boldsymbol{\eta} \quad \text{s.t.} \quad A \boldsymbol{\eta} \geq \mathbf{0} : \boldsymbol{\theta}, \quad (44)$$

where  $\boldsymbol{\theta} \in \mathbb{R}^{N(N-1)}$  is the vector of dual variables. Note that for appropriately chosen  $c \in \mathbb{R}^{N(n+1)}$ , (7) is a special case of (44). Let  $\mathbf{s} \in \mathbb{R}^{N(N-1)}$  represent the slack variables such that  $\mathbf{s} = [\mathbf{s}_\ell]_{\ell \in \mathcal{N}}$ , where  $\mathbf{s}_\ell = [\mathbf{s}_{\ell\ell'}]_{\ell' \in \mathcal{N} \setminus \{\ell\}} \in \mathbb{R}^{N-1}$ . Given some  $\tau > 0$ , the perturbed KKT system is given as

$$\begin{aligned} G \boldsymbol{\eta} - A^\top \boldsymbol{\theta} + c &= \mathbf{0}, \quad A \boldsymbol{\eta} - \mathbf{s} = \mathbf{0}, \quad \mathbf{s}_{\ell\ell'} \boldsymbol{\theta}_{\ell\ell'} = \tau, \quad (\ell, \ell') \in \mathcal{P}, \\ \mathbf{s} &\geq \mathbf{0}, \quad \boldsymbol{\theta} \geq \mathbf{0}. \end{aligned} \quad (45)$$

Instead of directly solving the KKT system (for  $\tau = 0$ ), the primal-dual path following IPM methods inexactly solve the perturbed KKT conditions as  $\tau \searrow 0$ . Given  $\tau > 0$  and some point  $(\boldsymbol{\eta}, \mathbf{s}, \boldsymbol{\theta})$  such that  $\mathbf{s} > 0$  and  $\boldsymbol{\theta} > 0$ , the major operation is to compute the Newton direction for the nonlinear equation system in (45) from the given point. The Newton direction can be computed by solving the following system

$$\begin{bmatrix} G & -A^\top \\ A & \Theta^{-1}S \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\eta} \\ \Delta\boldsymbol{\theta} \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p - \mathbf{s} + \tau\Theta^{-1}\mathbf{1} \end{bmatrix}, \quad (46)$$

and setting  $\Delta\mathbf{s} = A\Delta\boldsymbol{\eta} + r_p$ , where  $S = \text{diag}(\mathbf{s})$ ,  $\Theta = \text{diag}(\boldsymbol{\theta})$ ,  $r_p = A\boldsymbol{\eta} - \mathbf{s}$ ,  $r_d = G\boldsymbol{\eta} - A^\top\boldsymbol{\theta} + c$ . (46) implies that  $\Delta\boldsymbol{\eta}$  can be computed by solving

$$(G + A^\top S^{-1}\Theta A)\Delta\boldsymbol{\eta} = -r_d + A^\top S^{-1}\Theta(-r_p - \mathbf{s} + \tau\Theta^{-1}\mathbf{1}). \quad (47)$$

It is easy to see that  $M \triangleq G + A^\top S^{-1}\Theta A$  is indeed a block arrowhead matrix. Indeed, let  $\mathbf{d} = S^{-1}\Theta\mathbf{1} \in \mathbb{R}^{N(N-1)}$ , i.e.,  $\mathbf{d}_{\ell\ell'} = \boldsymbol{\theta}_{\ell\ell'}/\mathbf{s}_{\ell\ell'}$  for  $(\ell, \ell') \in \mathcal{P}$ , and define  $\mathbf{d}_\ell = [\mathbf{d}_{\ell\ell'}]_{\ell' \in \mathcal{N} \setminus \{\ell\}} \in \mathbb{R}^{N-1}$  for each  $\ell \in \mathcal{N}$ . Since  $A = [A_1 A_2]$ , from the definition of  $A_1$  and  $A_2$  in (6), it follows that  $M$  can be written as

$$M = \begin{pmatrix} M_{00} & M_{01} & M_{02} & \cdots & M_{0N} \\ M_{01}^\top & M_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ M_{02}^\top & \mathbf{0}^\top & M_{22} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{0N}^\top & \mathbf{0}^\top & \mathbf{0}^\top & \cdots & M_{NN} \end{pmatrix} \in \mathbb{R}^{N(n+1) \times N(n+1)}, \quad \text{where} \quad (48)$$

$$M_{00} = \mathbf{I}_N + \sum_{\ell \in \mathcal{N}} T_\ell^\top \text{diag}(\mathbf{d}_\ell) T_\ell, \quad M_{\ell\ell} = \gamma \mathbf{I}_n + X_\ell^\top \text{diag}(\mathbf{d}_\ell) X_\ell, \quad M_{0\ell} = T_\ell^\top \text{diag}(\mathbf{d}_\ell) X_\ell,$$

for  $\ell \in \mathcal{N}$ . Define  $R_\ell \triangleq T_\ell^\top \text{diag}(\mathbf{d}_\ell) T_\ell$  for  $\ell \in \mathcal{N}$ . Since  $X_\ell = -T_\ell \bar{X}$ , we have

$$M_{00} = \mathbf{I}_N + \sum_{\ell \in \mathcal{N}} R_\ell, \quad M_{\ell\ell} = \gamma \mathbf{I}_n + \bar{X}^\top R_\ell \bar{X}, \quad M_{0\ell} = -R_\ell \bar{X}, \quad \ell \in \mathcal{N}. \quad (49)$$

Moreover, due to structure of  $T_\ell$  (see Definition 1),  $R_\ell$  is a symmetric sparse matrix with a very special structure. In particular, it has only  $3N - 2$  nonzero elements, and  $R_\ell X$  can be computed in  $\mathcal{O}(Nn)$  flops. Hence, forming  $M_{\ell\ell}$  and  $M_{0\ell}$  require  $\mathcal{O}(Nn(n+1))$  and  $\mathcal{O}(Nn)$  flops, respectively. It is easy to show that forming  $M_{00}$  can be done in  $\mathcal{O}(N^2)$  flops; therefore, constructing  $M$  requires  $\mathcal{O}(N^2n(n+2))$  flops in total.

In the next lemma, we show that given an arbitrary  $b \in \mathbb{R}^{N(n+1)}$ , the solution to the system  $M\Delta\boldsymbol{\eta} = b$  for  $M$  given in (48) can be directly computed as in (50). Alternatively, one can also compute the Cholesky factorization of  $M$  first, and then use forward-backward substitution to compute the solution, which requires roughly the same amount of work that computing (50) requires. In the proof of Theorem 4, we also show as a side result that the Cholesky factorization of a generic block arrowhead matrix as in (48) can be computed very efficiently, compared to factorization of a dense matrix.

**Theorem 4** Let  $M \in \mathbb{R}^{N(n+1) \times N(n+1)}$  be a symmetric positive definite matrix with the generic block arrowhead structure given as in (48), where  $M_{00} \in \mathbb{R}^{N \times N}$ ,  $M_{0\ell} \in \mathbb{R}^{N \times n}$  and  $M_{\ell\ell} \in \mathbb{R}^{n \times n}$  for  $\ell \in \mathcal{N}$ . Given arbitrary  $b \in \mathbb{R}^{N(n+1)}$  such that  $b^\top = [b_0^\top b_1^\top \cdots b_N^\top]^\top$ , the system  $M\Delta\eta = b$  can be efficiently solved requiring  $\mathcal{O}(N^3 + N^2n^2 + 2Nn^3)$  flops, where  $b_0 \in \mathbb{R}^N$ ,  $b_\ell \in \mathbb{R}^n$  for  $\ell \in \mathcal{N}$ , and  $\Delta\eta^\top = [\Delta\mathbf{y}^\top \Delta\xi_1^\top \cdots \Delta\xi_N^\top]$ . The solution is given as

$$\Delta\mathbf{y} = M_{00}^{-1} \left( b_0 - \sum_{\ell \in \mathcal{N}} M_{0\ell} M_{\ell\ell}^{-1} b_\ell \right), \quad \Delta\xi_\ell = M_{\ell\ell}^{-1} (b_\ell - M_{0\ell}^\top \Delta\mathbf{y}), \quad \ell \in \mathcal{N}. \quad (50)$$

*Proof* In order to compute the Cholesky decomposition, we appropriately permute  $M$  and consider the following equation system:

$$\begin{pmatrix} M_{11} & \mathbf{0} & \cdots & \mathbf{0} & M_{01}^\top \\ \mathbf{0}^\top & M_{22} & \cdots & \mathbf{0} & M_{02}^\top \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}^\top & \mathbf{0}^\top & \cdots & M_{NN} & M_{0N}^\top \\ M_{01} & M_{02} & \cdots & M_{0N} & M_{00} \end{pmatrix} \begin{pmatrix} \Delta\xi_1 \\ \vdots \\ \Delta\xi_N \\ \Delta\mathbf{y} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \\ b_0 \end{pmatrix} \quad (51)$$

Let  $M_{\text{per}}$  be the matrix on the left hand side of (51). Compared to  $M$ , Cholesky decomposition of  $M_{\text{per}}$  can be computed much more efficiently. Indeed, diagonal blocks are factorized first:  $M_{00} = F_0 F_0^\top$ , and  $M_{\ell\ell} = F_\ell F_\ell^\top$  for  $\ell \in \mathcal{N}$ . Since  $M$  is positive definite, all the blocks on the diagonal are also positive definite; hence,  $F_0$  and  $F_\ell$  for  $\ell \in \mathcal{N}$  are invertible. The Cholesky factorization  $M_{\text{per}} = L_{\text{per}} L_{\text{per}}^\top$  can be easily verified:

$$L_{\text{per}} = \begin{pmatrix} F_1 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0}^\top & F_2 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0}^\top & \mathbf{0}^\top & \cdots & F_N & \mathbf{0} \\ L_1 & L_2 & \cdots & L_N & F_0 \end{pmatrix}, \quad (52)$$

where  $L_\ell = M_{0\ell} (F_\ell^{-1})^\top$  for  $\ell \in \mathcal{N}$ . Note that Cholesky factorization of  $M_{\ell\ell}$  can be computed with  $\mathcal{O}(n^3)$  complexity for each  $\ell \in \mathcal{N}$ , and with  $\mathcal{O}(N^3)$  for  $\ell = 0$ . Hence, the total complexity for computing  $\{F_\ell\}_{\ell \in \mathcal{N} \cup \{0\}}$  is  $\mathcal{O}(N^3 + Nn^3)$ . Moreover, for each  $\ell \in \mathcal{N}$ , computing  $L_\ell$  requires  $\mathcal{O}(n^3)$  flops for inverting the lower diagonal matrix  $F_\ell$ , and  $\mathcal{O}(Nn^2)$  for the multiplication; thus, the total complexity of computing  $L_{\text{per}}$  is  $\mathcal{O}(N^3 + N^2n^2 + 2Nn^3)$ . Moreover, storing  $L_{\text{per}}$  requires roughly  $N(n^2 + N)/2 + N^2n$  memory locations. Finally, computing  $\Delta\eta$  requires one forward and one backward substitution which will roughly add another  $\mathcal{O}(Nn^2 + N^2n)$  flops to the complexity.

Instead computing Cholesky factorization  $M_{\text{per}} = L_{\text{per}} L_{\text{per}}^\top$  explicitly, we will derive a closed form update rule for  $\Delta\eta$ . This will save us from storing  $L_{\text{per}}$  and from doing additional forward-backward substitutions. First, we solve  $L_{\text{per}} \Delta r = b$  via forward substitution, where  $\Delta r^\top = [\Delta r_1^\top \cdots \Delta r_N^\top \Delta r_0^\top]^\top$ . From (52), it clearly follows that

$$\Delta r_0 = F_0^{-1} (b_0 - \sum_{\ell \in \mathcal{N}} L_\ell \Delta r_\ell), \quad \Delta r_\ell = F_\ell^{-1} b_\ell, \quad \ell \in \mathcal{N}. \quad (53)$$

Next, we solve  $L_{\text{per}}^\top \Delta \boldsymbol{\eta} = \Delta r$  for  $\Delta \boldsymbol{\eta}$  via backward substitution:

$$\Delta \mathbf{y} = (F_0^\top)^{-1} \Delta r_0, \quad \Delta \boldsymbol{\xi}_\ell = (F_\ell^\top)^{-1} (\Delta r_\ell - L_\ell^\top \Delta \mathbf{y}), \quad \ell \in \mathcal{N}. \quad (54)$$

Note that for each  $\ell \in \mathcal{N}$ , from the definitions of  $F_\ell$  and  $L_\ell$ , it follows that

$$L_\ell F_\ell^{-1} = M_{0\ell} (F_\ell^{-1})^\top F_\ell^{-1} = M_{0\ell} M_{\ell\ell}^{-1}. \quad (55)$$

Therefore, using (53), (54), and (55), we can solve for  $\Delta \boldsymbol{\eta}$  in closed form as shown in (50).  $\square$

As we discussed before, the bottleneck step while solving (7) using a primal-dual path following IPM is to solve either the augmented system in (46) or the normal equations in (47). This reduces to computing the Cholesky decomposition of  $M$  in (48) with components defined in (49) and using forward-backward substitution to compute  $\Delta \boldsymbol{\eta}$ . Alternatively, according to Theorem 4, one can also directly compute the solution as in (50). Both alternatives have roughly the same complexity requiring  $\mathcal{O}(N^3 + N^2 n^2 + 2Nn^3)$  flops. Clearly, when  $N$  is large, i.e.,  $N \geq 10^5$ , this bottleneck step becomes impractical. On the other hand, combining P-APG and IPM, leaves the form of the bottleneck step unchanged, while making it more manageable by dividing it into smaller subsystem solves. In particular, the total complexity of computing Step 1 in P-APG consists of the complexity of solving  $K$  small size QPs as shown in (13).

Consider the problem in Step 1 of P-APG, and let  $c = -C^\top \tilde{\boldsymbol{\theta}}^k \in \mathbb{R}^{N(n+1)}$ . For each  $i \in \mathcal{K}$ , define  $c_i \in \mathbb{R}^{\bar{N}(n+1)}$  such that  $c_i$  is the subvector of  $c$  corresponding to the indices of  $\boldsymbol{\eta}_i = [\mathbf{y}_i^\top \boldsymbol{\xi}_i^\top]^\top$ , i.e.,  $\langle c, \boldsymbol{\eta} \rangle = \sum_{i \in \mathcal{K}} \langle c_i, \boldsymbol{\eta}_i \rangle$  for any  $\boldsymbol{\eta}$ . Moreover, let  $\bar{G} = \begin{pmatrix} \mathbf{I}_{\bar{N}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{\bar{N}n} \end{pmatrix}$ , and  $\bar{A}^i = [\bar{A}_1^{ii} \ \bar{A}_2^{ii}]$  where  $\bar{A}_1^{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1) \times \bar{N}}$  and  $\bar{A}_2^{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1) \times \bar{N}n}$  are defined in Definition 2. Hence, the problem in Step 1 of Fig. 1 can be equivalently written as

$$\min_{\boldsymbol{\eta}_i} \frac{1}{2} \boldsymbol{\eta}_i^\top \bar{G} \boldsymbol{\eta}_i + c_i^\top \boldsymbol{\eta}_i \quad \text{s.t.} \quad \bar{A}^i \boldsymbol{\eta}_i \geq 0 : \boldsymbol{\theta}_{ii}, \quad i \in \mathcal{K}, \quad (56)$$

where  $\boldsymbol{\theta}_{ii} \in \mathbb{R}^{\bar{N}(\bar{N}-1)}$  is the vector of dual variables. For each  $i \in \mathcal{K}$ , (56) is in a similar form with the QP in (7). Therefore, we immediately have the following result as a corollary of Theorem 4.

**Corollary 2** *For each  $i \in \mathcal{K}$ , the normal equations corresponding to the QP in (56) are in the same form with (47) leading to a system with a block-arrowhead matrix as in (48) with much smaller dimensions. Thus, Newton direction computations require  $\mathcal{O}(\bar{N}^3 + \bar{N}^2 n^2 + 2\bar{N}n^3)$  flops for each  $i \in \mathcal{K}$ .*

Suppose that we have  $K$  parallel processors. It is worth noting that thanks to the separability of the problem in Step 1 of P-APG, i.e., (56), one can do this computation in parallel, running a primal-dual path following IPM on each one of the  $K$  processors, or sequentially running the primal-dual path following IPM on a single processor  $K$  times.

*Remark 2* The total number of IPM iterations until P-APG terminates can be analyzed using the iteration complexity results on inexact accelerated proximal gradient algorithms [34], where Schmidt et al. analyzed APG in Fig. 1 when  $\nabla\rho$  in Step 1 is computed inexactly. In particular, one does not need to solve QP-subproblems exactly in each P-APG iteration. Given a tolerance sequence  $\{\tau_k\} \subset \mathbb{R}_{++}$  such that  $\tau_k \searrow 0$ , the number of primal path-following IPM iterations to compute a  $\tau_k$ -optimal solutions to QP-subproblems in the  $k$ -th iteration of P-APG is bounded above by  $\mathcal{O}(\bar{N} \ln(\frac{1}{\tau_k}))$  – see Section 4.3.2 in [29] (similar bounds can be driven for primal-dual path-following IPMs as well). Moreover, since QP-subproblems are strongly convex,  $\tau_k$ -optimality in function values implies an error bound on gradient evaluations in Step 2 of P-APG.

Recall that under Assumption 1, we have  $N = K\bar{N}$  such that  $\bar{N} > n + 1$ . Below we consider the bottleneck memory requirement for solving (7) in 2 cases: running **a)** P-APG with a primal-dual IPM computing Step-1 in Fig. 2, and **b)** IPM *alone* on (7). For case **a)**, the memory bottleneck in each iteration is due to solution of  $K$  Newton systems corresponding to (56); on the other hand, for case **b)**, the memory bottleneck is due to solution of a much larger Newton system using the normal equations in (47). In a naive implementation of case **b)**, one stores the non-zero components of the Cholesky factor  $L_{\text{per}}$  in (52) corresponding to the block arrowhead matrix in (48) after permuting as in (51), which requires storing  $\mathcal{O}(N^2(n+1) + Nn^2) = K\mathcal{O}(K\bar{N}^2(n+1) + \bar{N}n^2)$  entries; while for case **a)**, for each  $i \in \mathcal{K}$ , one stores the non-zero components of a Cholesky factor, analogous to (52), for the QP in (56) – see Corollary 2; hence, this naive implementation requires storing  $K\mathcal{O}(\bar{N}^2(n+1) + \bar{N}n^2)$  entries for all the Cholesky factors in total, in addition to storing  $\bar{N}^2K(K-1)$  dual variables, i.e.,  $\boldsymbol{\theta} = [\boldsymbol{\theta}_{ij}]_{(i,j) \in \mathcal{G}} \in \mathbb{R}^{\bar{N}^2K(K-1)}$ . Furthermore, for case **b)**, in a more memory efficient implementation, (50) in Theorem 4 implies that  $\Delta\xi_\ell$  can be computed sequentially after computing  $\Delta\mathbf{y}$ , which requires to store  $\mathcal{O}(N^2)$  at any time at the expense of forming  $M_{\ell\ell}$  and  $M_{0\ell}$  twice. Similarly, one can exploit this fact for case **a)** as well while solving normal equations for each  $i \in \mathcal{N}$ , which requires  $\mathcal{O}(K\bar{N}^2)$  memory in total if  $K$  processors run in parallel, and  $\mathcal{O}(\bar{N}^2)$  if  $K$  QPs in (56) are solved sequentially on a single processor. Therefore, running IPM within P-APG reduces the memory requirement significantly at least by a factor of  $K$  in comparison to running IPM alone, e.g., if we partition  $N$  observations into  $K = 10$  subsets and each subproblem requires 1GB of memory, then running IPM alone requires roughly 100GB, while IPM within P-APG requires only 10GB in total. This discussion is summarized in Table 1. Finally, recall the discussion at the beginning of Section 2.5: neither  $A_1$  nor  $A_2$  needs to be stored in the memory; storing only  $\{\bar{x}_\ell\}_{\ell=1}^N$  is sufficient to be able to compute matrix-vector multiplications with  $A_1$  and  $A_2$ .

### 3 Competitive Methods

In this section, we discuss an active set method for solving (7), and a multi-block ADMM method recently proposed by [25] to solve problem (4).

Table 1: Comparison of Memory Usage

	IPM alone	P-APG with IPM
Naive	$\mathcal{O}(K^2 \bar{N}^2 n)$	$\mathcal{O}(K \bar{N}^2 (n + K))$
Memory Efficient	$\mathcal{O}(K^2 \bar{N}^2)$	$\begin{matrix} \mathcal{O}(K \bar{N}^2) & \text{parallel} \\ \mathcal{O}(\bar{N}^2) & \text{sequential} \end{matrix}$

### 3.1 Active Set Method (ASM)

Although the number of constraints is  $\mathcal{O}(N^2)$  in (7), one expects that only few of them will be potentially active at the optimal solution; furthermore, this indeed turned out to be the case based on our numerical results for the test problems we considered in this paper – the number of active constraints was roughly  $\mathcal{O}(N)$ . Therefore, in this section, we briefly state a primal active set method to solve the regularized convex regression problem in (7) as an immediate alternative to P-APG method, and compare it with our P-APG method. One issue with primal active set methods is to determine an initial feasible point such that only very few constraints are active; and usually to overcome this problem one can use either “Phase I” or “big M” techniques. However, as we have already seen in the proof of Lemma 2, it is easy to construct an interior point for the polyhedron in (7) in spite of  $\mathcal{O}(N^2)$  constraints defining the set. In particular, let  $\hat{x} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \bar{x}_\ell$  and  $\hat{y} \triangleq \frac{1}{N} \sum_{\ell \in \mathcal{N}} \bar{y}_\ell$ , and for an arbitrary  $\alpha > 0$ , define  $\tilde{\mathbf{y}} = [\tilde{y}_\ell]_{\ell \in \mathcal{N}}$  and  $\tilde{\boldsymbol{\xi}} = [\tilde{\xi}_\ell]_{\ell \in \mathcal{N}}$  such that

$$\tilde{y}_\ell \triangleq \hat{y} + \frac{\alpha}{2} \|\bar{x}_\ell - \hat{x}\|_2^2, \quad \tilde{\xi}_\ell \triangleq \alpha(\bar{x}_\ell - \hat{x}), \quad \ell \in \mathcal{N}. \quad (57)$$

According to (27),  $\tilde{\boldsymbol{\eta}} \triangleq [\tilde{\mathbf{y}}^\top \tilde{\boldsymbol{\xi}}^\top]^\top$  is a Slater point such that  $A\tilde{\boldsymbol{\eta}} = A_1\tilde{\mathbf{y}} + A_2\tilde{\boldsymbol{\xi}} \geq \frac{\alpha\nu}{2}\mathbf{1}$ , where  $A = [A_1 A_2]$  and  $\nu > 0$  is defined in Lemma 2. Hence, no constraint is active at  $\tilde{\boldsymbol{\eta}}$ .

Consider (7), which can be restated in a more compact form: the QP in (44) with  $c = [\tilde{\mathbf{y}}^\top \mathbf{0}^\top]^\top$ . We will show that the primal active set algorithm shown in Fig. 3 can be efficiently implemented. In the rest,  $\mathbf{a}_{\ell\ell'}^\top$  denotes the row of  $A$  corresponding to index  $(\ell, \ell') \in \mathcal{P}$  – recall that the rows of  $A$  are sorted according to increasing lexicographic order on the index set  $\mathcal{P}$ .

**Definition 4** For  $k \geq 1$ , let  $\mathcal{W}^k \subset \mathcal{P}$  denote the working set at iteration  $k$ , which is a subset of active constraint indices, i.e.,  $\mathbf{a}_{\ell\ell'}^\top \boldsymbol{\eta}^k = 0$  for  $(\ell, \ell') \in \mathcal{W}^k$ , and  $m_k = |\mathcal{W}^k|$ . We form  $A^k = [\mathbf{a}_{\ell\ell'}^\top]_{(\ell, \ell') \in \mathcal{W}^k} \in \mathbb{R}^{m_k \times N(n+1)}$  concatenating the rows vertically, and define  $A_1^k \in \mathbb{R}^{m_k \times N}$  and  $A_2^k \in \mathbb{R}^{m_k \times Nn}$  as the submatrices of  $A^k$  such that  $A_1^k$  and  $A_2^k$  consist of columns of  $A^k$  corresponding to  $\mathbf{y}$  and  $\boldsymbol{\xi}$ , respectively.

The working set update strategy given in Fig. 3 ensures that  $\{\mathbf{a}_{\ell\ell'}\}_{(\ell, \ell') \in \mathcal{W}^k}$  are *linearly independent* for all  $k \geq 1$  – see [33] for details on this property, which we assume in the rest of this section.

**Algorithm ASM**Iteration 0: Set  $\boldsymbol{\eta}^0 \leftarrow \bar{\boldsymbol{\eta}}$  as in (57), and  $\mathcal{W}^0 \leftarrow \emptyset$ Iteration  $k$ : ( $k \geq 0$ )

- 1:  $\Delta\boldsymbol{\eta}^k \leftarrow \arg \min_{\Delta\boldsymbol{\eta}} \frac{1}{2} \Delta\boldsymbol{\eta}^\top G \Delta\boldsymbol{\eta} + (G\boldsymbol{\eta}^k + c)^\top \Delta\boldsymbol{\eta}$  s.t.  $\mathbf{a}_{\ell\ell'}^\top \Delta\boldsymbol{\eta} = 0 : \theta_{\ell\ell'}, (\ell, \ell') \in \mathcal{W}^k$
- 2: **if**  $\Delta\boldsymbol{\eta}^k = 0$ , **then**
- 3:     Compute  $\{\theta_{\ell\ell'}\}_{(\ell, \ell') \in \mathcal{W}^k} \subset \mathbb{R}$  such that  $\sum_{(\ell, \ell') \in \mathcal{W}^k} \mathbf{a}_{\ell\ell'} \theta_{\ell\ell'} = G\boldsymbol{\eta}^k + c$
- 4:     **if**  $\theta_{\ell\ell'} \geq 0$  for all  $(\ell, \ell') \in \mathcal{W}^k$ , **STOP** with solution  $\boldsymbol{\eta}^* = \boldsymbol{\eta}^k$ ;
- 5:     **else**  $(\bar{\ell}, \bar{\ell}') \leftarrow \arg \min_{(\ell, \ell') \in \mathcal{W}^k} \theta_{\ell\ell'}, \boldsymbol{\eta}^{k+1} \leftarrow \boldsymbol{\eta}^k, \mathcal{W}^{k+1} \leftarrow \mathcal{W}^k \setminus \{(\bar{\ell}, \bar{\ell}')\}$ ;
- 6: **else**  $\Delta\boldsymbol{\eta}^k \neq 0$
- 7:      $t^k \leftarrow \min \left\{ 1, \min \left\{ \frac{-\mathbf{a}_{\ell\ell'}^\top \boldsymbol{\eta}^k}{\mathbf{a}_{\ell\ell'}^\top \Delta\boldsymbol{\eta}^k} : (\ell, \ell') \notin \mathcal{W}^k \text{ s.t. } \mathbf{a}_{\ell\ell'}^\top \Delta\boldsymbol{\eta}^k < 0 \right\} \right\}$ ,
- 8:      $\boldsymbol{\eta}^{k+1} \leftarrow \boldsymbol{\eta}^k + t^k \Delta\boldsymbol{\eta}^k, \mathcal{I} \leftarrow \{(\ell, \ell') \notin \mathcal{W}^k : \mathbf{a}_{\ell\ell'}^\top \boldsymbol{\eta}^{k+1} = 0\}$
- 9:     **if**  $\mathcal{I} \neq \emptyset$ , **then**  $\mathcal{W}^{k+1} \leftarrow \mathcal{W}^k \cup \{(\ell, \ell')\}$  for some  $(\ell, \ell') \in \mathcal{I}$ ;
- 10:    **else** set  $\mathcal{W}^{k+1} \leftarrow \mathcal{W}^k$ ;

Fig. 3: Active Set Algorithm (ASM)

Note that in each iteration  $k \geq 1$ , we need to solve a subproblem to determine the direction  $\Delta\boldsymbol{\eta}^k$  as follows

$$\Delta\boldsymbol{\eta}^k = \arg \min_{\Delta\boldsymbol{\eta}} \frac{1}{2} \Delta\boldsymbol{\eta}^\top G \Delta\boldsymbol{\eta} + (G\boldsymbol{\eta}^k + c)^\top \Delta\boldsymbol{\eta} \quad \text{s.t.} \quad A^k \Delta\boldsymbol{\eta} = 0 : \boldsymbol{\theta}^k, \quad (58)$$

where  $\boldsymbol{\theta}^k = [\theta_{\ell\ell'}^k]_{(\ell, \ell') \in \mathcal{W}^k} \in \mathbb{R}^{m_k}$  denotes an optimal dual solution. Hence,  $(\Delta\boldsymbol{\eta}^k, \boldsymbol{\theta}^k)$  satisfies the KKT system corresponding to (58):

$$\begin{bmatrix} G & A^{k\top} \\ A^k & \mathbf{0} \end{bmatrix} \begin{bmatrix} -\Delta\boldsymbol{\eta}^k \\ \boldsymbol{\theta}^k \end{bmatrix} = \begin{bmatrix} G\boldsymbol{\eta}^k + c \\ 0 \end{bmatrix} \quad \Rightarrow \quad A^k G^{-1} A^{k\top} \boldsymbol{\theta}^k = A^k (\boldsymbol{\eta}^k + c) \quad (59)$$

since  $G^{-1}c = c$ . Therefore,  $\boldsymbol{\theta}^k$  can be computed via forward and backward substitution after computing the Cholesky factorization of  $A^k G^{-1} A^{k\top}$ ; next one can compute  $\Delta\boldsymbol{\eta}^k$  according to the first row in the KKT system as follows:  $\Delta\boldsymbol{\eta}^k = G^{-1} A^{k\top} \boldsymbol{\theta}^k - (\boldsymbol{\eta}^k + c)$ .

*Remark 3* It is worth noting that  $A^k G^{-1} A^{k\top} = A_1^k A_1^{k\top} + \frac{1}{\gamma} A_2^k A_2^{k\top}$ . For any  $k \geq 1$ ,  $A_1^k A_1^{k\top}$  and  $A_2^k A_2^{k\top}$  computations require  $\mathcal{O}(m_k^2)$  and  $\mathcal{O}(m_k^2 n)$  flops, respectively; and given  $\mathbf{a}_{\ell\ell'}$  for some  $(\ell, \ell') \in \mathcal{P} \setminus \mathcal{W}^k$ ,  $A^k G^{-1} \mathbf{a}_{\ell\ell'}$  can be computed in  $\mathcal{O}(m_k(n+1))$  flops. These complexity bounds can be easily verified by observing the structure in  $A^k$  after ordering its rows according to increasing lexicographic order on the index set  $\mathcal{W}^k$ .

Naively, the majority of total computational complexity at iteration  $k$  is mainly due to forming  $A^k G^{-1} A^{k\top} = A_1^k A_1^{k\top} + \frac{1}{\gamma} A_2^k A_2^{k\top}$  in  $\mathcal{O}(m_k^2(n+1))$  flops, and computing its Cholesky factorization in  $\mathcal{O}(m_k^3)$  flops – the factorization exists since  $A^k G^{-1} A^{k\top}$  is positive definite due to  $\text{rank}(A^k) = m_k$ . That said, at the end of each iteration the working set changes by at most one index; thus, one does not need to compute Cholesky factorizations from scratch. In particular,

because at most one row (constraint) is added or deleted from  $A^k$ , Cholesky factorization for  $A^{k+1}G^{-1}A^{k+1\top}$  can be updated very efficiently by using  $A^kG^{-1}A^{k\top} = L^kL^{k\top}$  from the previous iteration. Also, note  $\theta^k$  is a byproduct of this approach, so we don't need to compute it again in the following step if  $\eta^k = 0$ . Next, we will briefly discuss how to utilize the information from the previous iteration to solve the subproblems much more efficiently.

**Lemma 5** *For some  $m \geq 1$ , let  $B \in \mathbb{R}^{m \times N(n+1)}$  and  $b \in \mathbb{R}^{N(n+1)}$  such that  $\text{rank}(B) = m$  and  $b$  is not in the row-space of  $B$ . Suppose  $LL^\top$  represent the Cholesky factorization of  $BG^{-1}B^\top$  for some symmetric positive definite matrix  $G$ . Define  $\bar{B} = \begin{bmatrix} B \\ b^\top \end{bmatrix} \in \mathbb{R}^{(m+1) \times N(n+1)}$ . Then given  $L$ , Cholesky factorization for  $\bar{B}G^{-1}\bar{B}^\top = \bar{L}\bar{L}^\top$  can be computed as*

$$\bar{L} = \begin{bmatrix} L & \mathbf{0} \\ h^\top & d \end{bmatrix}, \quad h = L^{-1}BG^{-1}b, \quad d = \sqrt{b^\top G^{-1}b}. \quad (60)$$

*Proof* Since  $\text{rank}(\bar{B}) = m + 1$ , trivially  $\bar{B}G^{-1}\bar{B}^\top$  is positive definite, and it has a Cholesky factorization  $\bar{L}\bar{L}^\top$ . Moreover, it is easy to verify that  $\bar{L}$  given in (60) is the Cholesky factor.  $\square$

Assume that we already know Cholesky factorization  $A^kG^{-1}A^{k\top} = L^kL^{k\top}$ , and  $\mathcal{W}^{k+1} = \mathcal{W}^k \cup \{(\ell, \ell')\}$  for some  $(\ell, \ell') \in \mathcal{P} \setminus \mathcal{W}^k$ . Suppose  $\mathbf{a}_{\ell\ell'}^\top$  is appended to  $A^k$  as the last row to form  $A^{k+1}$ . Since  $\text{rank}(A^{k+1}) = m_k + 1$ , setting  $B = A^k$  and  $b = \mathbf{a}_{\ell\ell'}$  satisfies the conditions in Lemma 5. Thus, according to (60), the new factorization for  $A^{k+1}G^{-1}A^{k+1\top} = L^{k+1}L^{k+1\top}$  can be computed as  $L^{k+1} = \begin{bmatrix} L^k & \mathbf{0} \\ h^k & d^k \end{bmatrix}$ , which only requires to solve  $L^kh^k = A^kG^{-1}\mathbf{a}_{\ell\ell'}$  for  $h^k$ ,

and to compute  $d^k = \sqrt{\mathbf{a}_{\ell\ell'}^\top G^{-1}\mathbf{a}_{\ell\ell'}}$ . Note computing  $h^k$  requires forming  $A^kG^{-1}\mathbf{a}_{\ell\ell'}$ , which can be computed in  $\mathcal{O}(m_k n)$  flops according to Remark 3, and implementing one forward substitution, which can be done in  $\mathcal{O}(m_k^2)$  flops.

Now consider the case  $\mathcal{W}^{k+1} = \mathcal{W}^k \setminus \{(\ell, \ell')\}$  for some  $(\ell, \ell') \in \mathcal{W}^k$ . Note that  $\mathbf{a}_{\ell\ell'}^\top$  is an arbitrary row of  $A^k$  (not necessarily the last one). The following lemma will help us update the factorization corresponding to  $\mathcal{W}^{k+1}$  efficiently when we are given  $L^k$ .

**Lemma 6** *Let  $B_1 \in \mathbb{R}^{s_1 \times N(n+1)}$ ,  $B_2 \in \mathbb{R}^{s_2 \times N(n+1)}$ , and  $b \in \mathbb{R}^{N(n+1)}$  such that  $\text{rank}(B) = s_1 + s_2 + 1$ , where  $B = [B_1^\top \ b \ B_2^\top]^\top$ . Suppose  $LL^\top$  represent the Cholesky factorization of  $BG^{-1}B^\top$  for some symmetric positive definite matrix  $G$ , where  $L = \begin{bmatrix} L_1 & \mathbf{0}_{s_1 \times 1} & \mathbf{0}_{s_1 \times s_2} \\ h_1^\top & d & \mathbf{0}_{1 \times s_2} \\ F & h_2 & L_2 \end{bmatrix}$ . Define  $\bar{B} = [B_1^\top \ B_2^\top]^\top \in \mathbb{R}^{(s_1+s_2) \times N(n+1)}$ .*

*Then given  $L$ , Cholesky factorization for  $\bar{B}G^{-1}\bar{B}^\top = \bar{L}\bar{L}^\top$  can be computed as*

$$\bar{L} = \begin{bmatrix} L_1 & \mathbf{0}_{s_1 \times s_2} \\ F & \bar{L}_2 \end{bmatrix}, \quad \text{s.t.} \quad \bar{L}_2\bar{L}_2^\top = L_2L_2^\top + h_2h_2^\top. \quad (61)$$

Moreover, given  $L_2$  and  $h_2$ , computing  $\bar{L}_2$  requires  $\mathcal{O}(s_2^2)$  flops.



*Proof* It is easy to verify that  $\bar{L}$  given in (61) is the lower-triangular Cholesky factor of  $\bar{B}G^{-1}\bar{B}^\top$ . For details of computing  $\bar{L}_2$ , refer to [13]. Moreover, MATLAB routine `cholupdate`( $L_2^\top, h_2$ ) can be called to compute  $\bar{L}_2^\top$ .  $\square$

**Algorithm Multi-Block ADMM**

Iteration 0:  $\Delta_{ij} \leftarrow \bar{x}_i - \bar{x}_j$  for  $(i, j) \in \mathcal{N} \times \mathcal{N}$  and  $\bar{\Delta}_j \leftarrow (\sum_{i \in \mathcal{N}} \Delta_{ij} \Delta_{ij}^\top)^{-1}$  for  $j \in \mathcal{N}$   
 Iteration  $k$ : ( $k \geq 0$ )

- 1:  $\xi_j^{k+1} \leftarrow \bar{\Delta}_j (\sum_{i \in \mathcal{N}} \Delta_{ij} (\theta_{ij}^k / \rho + \nu_{ij}^k + y_i^k - y_j^k))$  for  $j \in \mathcal{N}$
- 2:  $\tilde{\nu}_{ij}^{k+1} \leftarrow \nu_{ij}^k - \Delta_{ij}^\top \xi_j^{k+1}$  for  $(i, j) \in \mathcal{N} \times \mathcal{N}$
- 3:  $\mathbf{w}^{k+1} \leftarrow \mathbf{y} + D^\top \boldsymbol{\theta}^k + \rho D^\top \tilde{\mathbf{v}}^{k+1}$
- 4:  $y_i^{k+1} \leftarrow \frac{1}{1+2N\rho} (w_i + 2\rho \sum_{j \in \mathcal{N}} w_j)$  for  $i \in \mathcal{N}$
- 5:  $\nu_{ij}^{k+1} \leftarrow \min \left\{ y_j^{k+1} + \Delta_{ij}^\top \xi_j^{k+1} - y_i^{k+1} - \theta_{ij}^k / \rho, 0 \right\}$  for  $(i, j) \in \mathcal{N} \times \mathcal{N}$
- 6:  $\theta_{ij}^{k+1} \leftarrow \theta_{ij}^k + \rho (\nu_{ij}^{k+1} + y_i^{k+1} - y_j^{k+1} - \Delta_{ij}^\top \xi_j^{k+1})$  for  $(i, j) \in \mathcal{N} \times \mathcal{N}$

Fig. 4: Multi-block ADMM (ADMM)

### 3.2 Multi-block ADMM

Recently, Mazumder et al. [25] proposed a multi-block ADMM to solve problem (4). Although, the authors report that it works well in practice, to our best knowledge, the convergence property of the method is still unknown. In fact, it is recently shown that ADMM does not necessarily converge when the number of primal variable blocks are three or more [9]; and the ADMM algorithm in [25], displayed in Fig. 4, alternately updates *three*-blocks of primal variables:  $\boldsymbol{\xi} = [\xi_i]_{i \in \mathcal{N}}$ ,  $\mathbf{y} = [y_i]_{i \in \mathcal{N}}$  and  $\boldsymbol{\nu} = [\nu_{ij}]_{(i,j) \in \mathcal{N} \times \mathcal{N}}$ .

The matrix  $D \in \mathbb{R}^{N^2 \times N}$  is similar to our matrix  $A_1$  defined in Definition 1, except  $D$  also contains rows corresponding to  $(i, i) \in \mathcal{N} \times \mathcal{N}$ , i.e.,  $D\mathbf{y} = \mathbf{z} \in \mathbb{R}^{N^2}$  such that  $z_{ij} = y_j - y_i$  for  $(i, j) \in \mathcal{N} \times \mathcal{N}$ , and the long-vector  $\mathbf{z}$  obtained by sorting its elements according to increasing lexicographic order on the index set  $\mathcal{N} \times \mathcal{N}$ . Similarly, the elements of the auxiliary variable  $\tilde{\mathbf{v}} \in \mathbb{R}^{N^2}$  is also sorted according to increasing lexicographic order on the index set  $\mathcal{N} \times \mathcal{N}$ . During initialization, the ADMM algorithm requires computing  $\bar{\Delta}_j = (\sum_{i \in \mathcal{N}} \Delta_{ij} \Delta_{ij}^\top)^{-1}$  for all  $j \in \mathcal{N}$ , where  $\Delta_{ij} = \bar{x}_i - \bar{x}_j$ . Although it is required only one time, this computation costs  $\mathcal{O}(N^2 n^2 + N n^3)$  flops. Based on our numerical tests, as  $N$  increases, this preprocessing time becomes substantial compared to overall runtime. At each iteration, the algorithm needs to update five different variables:  $\boldsymbol{\xi}^k$ ,  $\mathbf{w}^k$ ,  $\mathbf{y}^k$ ,  $\boldsymbol{\nu}^k$  and  $\boldsymbol{\theta}^k$ . The cost for updating subgradient vector  $\boldsymbol{\xi}^k$  is  $\mathcal{O}(N^2 n + N n^2)$  flops, updating  $\mathbf{w}^k$  takes  $\mathcal{O}(N^2)$  flops, and given  $\mathbf{w}^{k-1}$  updating the function value-vector  $\mathbf{y}^k$  takes  $\mathcal{O}(N)$  flops, and updating residuals  $\boldsymbol{\nu}^k$  and dual variables  $\boldsymbol{\theta}^k$  both take  $\mathcal{O}(N^2)$  flops separately. Thus, the overall per iteration complexity is  $\mathcal{O}(N^2 n + N n^2)$  with  $\mathcal{O}(N^2 n^2 + N n^3)$  one-time cost at the beginning. Note that ADMM needs to store not only matrix

$D \in \mathbb{R}^{N^2 \times N}$  and vectors  $\Delta_{ij} \in \mathbb{R}^n$  for all  $(i, j) \in \mathcal{N}$ , that are comparable to our  $A_1$  and  $A_2$ , but also  $\bar{\Delta}_i \in \mathbb{R}^{n \times n}$  for all  $i \in \mathcal{N}$ , which are the matrices inverted during pre-processing; hence, the number of non-zeros stored in the RAM for ADMM is roughly  $(N^2 - N)(n + 2) + Nn^2$ , which is  $\mathcal{O}(K^2 \bar{N}^2 n)$ . When compared to Table 1, clearly P-APG leads to significant memory savings.

#### 4 Numerical Study

Here we demonstrate the scalability of P-APG, and compare its performance against other competitive methods: an interior point method, an active set method (ASM), and a multi-block ADMM. To solve the convex regression problem, we implemented P-APG, ASM and ADMM in MATLAB, and used the stand-alone version MOSEK [27] as an interior point solver for benchmarking purposes. Moreover, for P-APG, we also use MOSEK together with the Parallel Computing Toolbox, in order to solve  $K$  QP-subproblems in *parallel* using  $K$  cores in each iteration of P-APG in Fig. 2. MOSEK is a commercial off-the-shelf software which has a state-of-the-art interior-point optimizer for quadratic problems. Note that MOSEK also comes with CVX, which is a popular MATLAB-based modeling system for convex optimization; but this version of MOSEK is not compatible with Parallel Computing Toolbox in MATLAB, i.e., even though one calls MOSEK through CVX formulations within a `parfor` loop, the  $K$  subproblems are still solved in a sequential manner. In order to take advantage of the computing power in a cluster of computers for long-running jobs, one has to adopt *batch* processing in MATLAB to be able to better exploit the processor cores in multiple machines. On the other hand, matrix operations in MATLAB leverage multi-core and multi-threading framework by default. Hence, ASM and ADMM are coded without using the parallel toolbox, as they only contain matrix operations in every iteration and these operations are executed in parallel automatically. To eliminate factors that might have an influence on the runtime to the best extent, we carried out all numerical tests comparing P-APG against other methods on high performance computing cluster by executing a single script, so that they all run on exactly the same processor cores and memory modules. Numerical tests are carried out on a single node at a research computing cluster. The node is composed of one 24-core processor, each having 1GB RAM (24GB RAM in total). We determine the number of core processors and the amount of RAM allocated depending on the size of the problem solved – see Sections 4.1 and 4.2.

*Experimental setup:* Our problem setup adopted in the following sections involve two different test functions: 1)  $f_0(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top Q\mathbf{x}$  and 2)  $f_0(\mathbf{x}) = \exp(\mathbf{p}^\top \mathbf{x})$ , where  $Q \in \mathbb{R}^{n \times n}$  is a symmetric matrix,  $\mathbf{p} \in \mathbb{R}^n$ , and they are randomly generated as follows. We first set  $\bar{Q} \triangleq \Lambda^\top \Lambda$  such that  $\Lambda \in \mathbb{R}^{n \times n}$  is generated randomly with all components being i.i.d. with  $\mathcal{N}(0, 1)$ , where  $\mathcal{N}(\mu, \sigma^2)$  denotes Normal distribution with mean  $\mu$  and variance  $\sigma^2$ ; next, without changing left and right singular vectors of  $\bar{Q}$ , we transform its singular values such that the resulting condition number is 15 and we call the resulting matrix as  $Q$ ; and  $\mathbf{p} \in \mathbb{R}^n$  is generated using uniform distribution on the hypercube  $[0, 0.2]^n$ . The noisy observations  $\{\bar{y}_\ell\}_{\ell \in \mathcal{N}}$  are generated according to (1), where the locations

$\{\bar{x}_\ell\}_{\ell=1}^N \subset \mathbb{R}^n$  and additive noise  $\{\epsilon_\ell\}_{\ell=1}^N \subset \mathbb{R}$  are generated randomly with all components being i.i.d. with  $\mathcal{N}(0, 4)$  and  $\mathcal{N}(0, 100)$  respectively. In addition, we moved 30% of randomly chosen location/observation pairs into the interior of the epigraph of the test function  $f_0$  by replacing  $(\bar{x}_\ell, \bar{y}_\ell)$  with  $(\bar{x}_\ell, 1.3\bar{y}_\ell)$ . In all the experiments involving P-APG and ASM, we set  $\gamma = 10^{-4}$  in (7).

#### 4.1 Convergence behavior of P-APG on the regularized problem

We compare i) running MOSEK alone and ii) running it within P-APG on the regularized problem (7) with increasing dimension. The numerical study is mainly aimed to demonstrate how the performance of each method scales for solving the *regularized* problem as its dimension increases. First, we start with a small size problem:  $n = 10$ ,  $N = 100$ , and use the test function  $f_0(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x}$ . We compare the quality of the solutions computed by P-APG and dual gradient ascent (as the dual function  $g_\gamma$  in (22) is differentiable). In order to compute dual gradient,  $\nabla g_\gamma$ , one needs to solve  $K$  quadratic subproblems. To exploit this parallel structure, we partition the data into two sets, i.e.,  $K = 2$ . Within both dual gradient ascent and P-APG, we called MOSEK to compute the dual gradients via solving  $K$  QP-subproblems. Since we allow violations for the relaxed constraints, we define the “duality gap” at the  $k$ -th iteration as  $\boldsymbol{\theta}_k^\top \mathbf{C}\boldsymbol{\eta}_k$  – recall that  $\boldsymbol{\eta}_k = [\mathbf{y}_k^\top \boldsymbol{\xi}_k^\top]^\top$ . Fig. 5(left) represents how the duality gap for both methods change at each iteration. In order to better understand the behavior of P-APG, we report in Fig. 5(right) the duality gap of P-APG in a larger scale. Fig. 6 reports the infeasibility of iterates, i.e.,  $\|(A_1 \mathbf{y}_k + A_2 \boldsymbol{\xi}_k)_-\|_2$ .

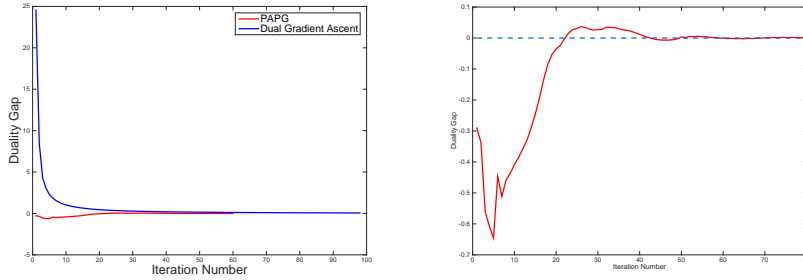


Fig. 5: Duality Gap for P-APG and Dual Gradient Ascent: (left) P-APG and Dual Gradient Ascent, (right) Zoom-in for P-APG Method

A primal-dual iterate  $(\boldsymbol{\eta}, \boldsymbol{\theta})$  is optimal if the duality gap and infeasibility are both zero. As the feasibility happens in the limit, the duality gap in Fig. 5(right) can go below 0, which can be explained by the infeasibility of iterates. Therefore, observing a decrease in duality gap only tells one part of the story; without convergence to feasibility, it is not valuable alone as a measure. As shown in the Fig. 5, the duality gap converges quickly to zero for both methods. On the other hand, as shown in Fig. 6, constraint violation for P-APG iterates decreases to

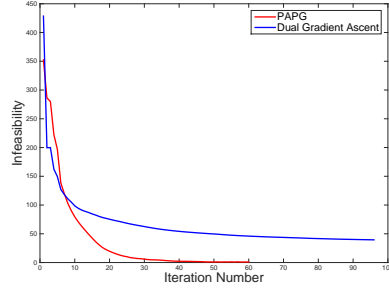


Fig. 6: Distance to Feasible Region for P-APG and Dual Gradient Ascent.

0 much faster than it does for the dual gradient ascent iterates. Hence, P-APG iterate sequence converges to the unique optimal solution considerably faster.

As shown in Tables 2, 3, 4 and 5, the dimension of variables  $n \in \{20, 80\}$ , and the number of observations  $N \in \{200, 400, 800, 1600, 2400\}$ . Since the number of constraints increases at the rate of  $\mathcal{O}(N^2)$ , as the size of problem increases in  $N$ , we reported the normalized infeasibility  $\|(A_1\mathbf{y} + A_2\boldsymbol{\xi})_-\|_2 / \sqrt{N^2 - N}$  and normalized duality gap  $|\boldsymbol{\theta}^\top C\boldsymbol{\eta}| / (N^2 - N)$ . We partition the set of observations  $\mathcal{N}$  into  $K$  subsets. Each one of them consists of 100 points; therefore, we set  $K = 2, 4, 8, 16, 24$  for  $N = 200, 400, 800, 1600, 2400$ , and we reserve  $2/2, 4/4, 8/8, 16/16, 24/24$  number of Cores/RAM, respectively, depending on  $N$  so that for each job submitted to the computing cluster, an instance of (7) is solved using P-APG on the node such that each subproblem in (56) for  $i \in \mathcal{K}$  is computed on a different core. We tested both the adaptive step and constant step version of P-APG, which we abbreviate as PAPG\_A and PAPG\_C, respectively. Both PAPG\_A and PAPG\_C are terminated whenever they compute a primal-dual iterate,  $\boldsymbol{\eta} = [\mathbf{y}^\top \boldsymbol{\xi}^\top]^\top$  and  $\boldsymbol{\theta}$ , satisfying the stopping criteria:  $\|(A_1\mathbf{y} + A_2\boldsymbol{\xi})_-\| / \sqrt{N^2 - N} \leq 1e-1$  and  $|\boldsymbol{\theta}^\top C\boldsymbol{\eta}| / (N^2 - N) \leq 5e-7$ , or at the end of 2 hours, which are reported as Infeasibility and DualGap respectively in the tables. Moreover, we also report relative suboptimality,

Table 2: Comparison with test function  $\frac{1}{2}\mathbf{x}^\top Q\mathbf{x}$  for  $n = 20$

n, N	Algorithm	Cores/RAM	Preprocess	Wall-time	Infeasibility	SubOpt_Reg	DualGap
20, 200	Mosek_Reg	2/2	0	2	0	0	0
	PAPG_A	2/2	0.2	30	9.51E-02	3.86E-04	6.80E-08
	PAPG_C	2/2	0.2	16	9.75E-02	1.22E-03	2.63E-07
20, 400	Mosek_Reg	4/4	0	9	0	0	0
	PAPG_A	4/4	1.1	47	9.54E-02	1.92E-03	2.56E-07
	PAPG_C	4/4	1.0	44	9.94E-02	7.45E-05	3.23E-08
20, 800	Mosek_Reg	8/8	0	42	0	0	0
	PAPG_A	8/8	4.3	92	9.92E-02	6.14E-03	4.79E-07
	PAPG_C	8/8	4.2	101	9.87E-02	2.06E-03	1.75E-07
20, 1600	Mosek_Reg	16/16	0	311	0	0	0
	PAPG_A	16/16	20.3	259	9.50E-02	1.10E-02	4.80E-07
	PAPG_C	16/16	20.0	408	9.99E-02	5.89E-03	2.63E-07
20, 2400	Mosek_Reg	24/24	0	987	0	0	0
	PAPG_A	24/24	56.1	323	9.77E-02	1.43E-02	4.61E-07
	PAPG_C	24/24	59.7	723	1.00E-01	7.42E-03	2.43E-07

Table 3: Comparison with test function  $\frac{1}{2}\mathbf{x}^\top Q\mathbf{x}$  for  $n = 80$ 

n, N	Algorithm	Cores/RAM	Preprocess	Wall-time	Infeasibility	SubOpt_Reg	DualGap
80, 200	Mosek_Reg	2/2	0	4	0	0	0
	PAPG_A	2/2	0.5	89	9.46E-02	6.08E-04	1.88E-07
	PAPG_C	2/2	0.5	54	9.93E-02	1.05E-03	3.28E-07
80, 400	Mosek_Reg	4/4	0	18	0	0	0
	PAPG_A	4/4	2.5	324	8.92E-02	5.79E-04	9.54E-08
	PAPG_C	4/4	2.6	287	9.87E-02	1.09E-03	1.96E-07
80, 800	Mosek_Reg	8/8	0	97	0	0	0
	PAPG_A	8/8	12.2	383	9.79E-02	1.43E-03	1.29E-07
	PAPG_C	8/8	12.5	379	9.92E-02	1.02E-03	9.12E-08
80, 1600	Mosek_Reg	16/16	0	661	0	0	0
	PAPG_A	16/16	55.9	597	9.59E-02	1.61E-03	7.67E-08
	PAPG_C	16/16	57.5	1145	9.94E-02	1.61E-03	7.70E-08
80, 2400	Mosek_Reg	24/24	0	1966	0	0	0
	PAPG_A	24/24	128.5	897	9.91E-02	1.17E-03	3.87E-08
	PAPG_C	24/24	133.8	1947	1.00E-01	2.57E-03	9.19E-08

Table 4: Comparison with test function  $\exp(\mathbf{p}^\top \mathbf{x})$  for  $n = 20$ 

n, N	Algorithm	Cores/RAM	Preprocess	Wall-time	Infeasibility	SubOpt_Reg	DualGap
20, 200	Mosek_Reg	2/2	0	3	0	0	0
	PAPG_A	2/2	0.2	35	8.97E-02	1.01E-03	3.58E-07
	PAPG_C	2/2	0.2	17	9.62E-02	1.16E-03	4.09E-07
20, 400	Mosek_Reg	4/4	0	10	0	NaN	0
	PAPG_A	4/4	1.0	67	9.41E-02	1.36E-03	2.15E-07
	PAPG_C	4/4	1.0	71	9.93E-02	6.50E-04	7.11E-08
20, 800	Mosek_Reg	8/8	0	66	0	0	0
	PAPG_A	8/8	4.2	194	7.04E-02	5.56E-03	4.62E-07
	PAPG_C	8/8	4.1	266	9.91E-02	5.39E-03	4.45E-07
20, 1600	Mosek_Reg	16/16	0	558	0	0	0
	PAPG_A	16/16	18.7	553	9.75E-02	6.14E-02	3.66E-07
	PAPG_C	16/16	18.8	886	9.72E-02	1.65E-03	4.51E-07
20, 2400	Mosek_Reg	24/24	0	2155	0	0	0
	PAPG_A	24/24	128.5	797	9.97E-02	5.79E-02	1.29E-07
	PAPG_C	24/24	133.8	1347	1.00E-01	2.92E-03	3.59E-08

Table 5: Comparison with test function  $\exp(\mathbf{p}^\top \mathbf{x})$  for  $n = 80$ 

n, N	Algorithm	Cores/RAM	Preprocess	Walltime	Infeasibility	SubOpt_Reg	DualGap
80, 200	Mosek_Reg	2/2	0	4	0	0	0
	PAPG_A	2/2	0.4	53	9.21E-02	6.68E-03	4.25E-07
	PAPG_C	2/2	0.4	34	9.27E-02	1.60E-03	1.01E-07
80, 400	Mosek_Reg	4/4	0	21	0	0	0
	PAPG_A	4/4	2.6	209	9.39E-02	3.00E-03	1.15E-07
	PAPG_C	4/4	2.6	183	9.63E-02	2.92E-03	1.12E-07
80, 800	Mosek_Reg	8/8	0	143	0	0	0
	PAPG_A	8/8	12.8	299	9.12E-02	9.44E-04	1.90E-08
	PAPG_C	8/8	13.0	272	9.87E-02	1.65E-04	5.01E-10
80, 1600	Mosek_Reg	16/16	0	687	0	0	0
	PAPG_A	16/16	55.3	465	9.72E-02	3.07E-03	3.59E-08
	PAPG_C	16/16	54.7	753	9.95E-02	1.28E-04	1.53E-09
80, 2400	Mosek_Reg	24/24	0	2789	0	0	0
	PAPG_A	24/24	138.2	774	9.69E-02	7.82E-03	6.39E-08
	PAPG_C	24/24	139.3	1379	9.94E-02	2.23E-03	2.01E-08

i.e.,  $\text{SubOpt\_Reg} = |p - p_\gamma^*|/p_\gamma^*$ , where  $p_\gamma^*$  denotes the optimal value to (7) and  $p$  denotes the objective value of (7) at termination. **Preprocess** for P-APG method is the wall-clock time elapsed during the computation of the maximum singular value for the matrix  $A_4$ . Additionally, in all the tables, N/A means that the wall clock time exceeded 2 hours for the job, and **Wall-time** stands for wall-clock time in *seconds* for the whole job including **Preprocess**.

The numerical results reported in Tables 2, 3, 4 and 5 show that P-APG solution is very close to the optimal solution of the regularized problem in (7). Note that MOSEK using interior point optimizer starts working slowly beyond  $N = 1600$  due to  $\mathcal{O}(N^2n)$  memory requirement – see Table 1. Numerical results show that advantages of P-APG over running IPM alone on (7) become more and more evident as the number of observations,  $N$ , increases.

#### 4.2 Comparison with ASM and ADMM

In this section, we compare P-APG with ASM and multi-block ADMM. It is worth noting that multi-block ADMM solves the original problem (3), while P-APG and the active set method (ASM) solve the regularized problem (7). All algorithms are terminated whenever they compute an iterate,  $(\mathbf{y}, \boldsymbol{\xi})$ , satisfying the following stopping criteria:  $\|\mathbf{y} - \mathbf{y}^*\|/\sqrt{N} \leq 5e-3$  and  $\|(A_1\mathbf{y} + A_2\boldsymbol{\xi})_-\|/\sqrt{N^2 - N} \leq 1e-1$ , where the first one is the relative suboptimality with respect to the original problem in (3) and the second one is the normalized infeasibility. The initial point for ASM is set by using (57), where  $\alpha = 1/N$ . This choice of  $\alpha$  works consistently well based on our test.

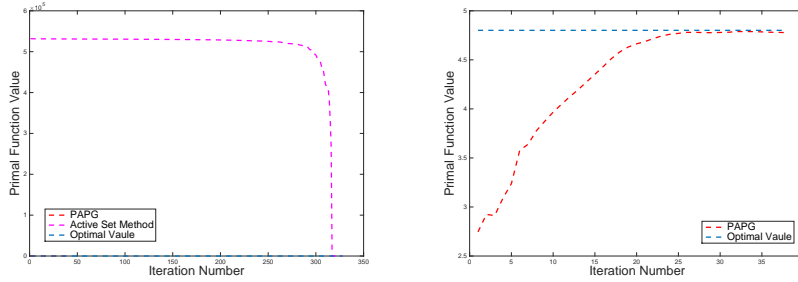


Fig. 7: P-APG vs ASM  $(n, N) = (10, 100)$ : (left) Relative Suboptimality, (right) Zoom-in for P-APG method

Experiments comparing convergence behaviors of P-APG and ASM on a small-size problem were carried out for  $(n, N) = (10, 100)$ . As shown in Fig. 7, active set algorithm spends quite long in a *warm-up* phase before making noticeable progress in terms of function value; and this behavior becomes more and more apparent as the size of the problem increases. Fig. 8(left) displays how the number of active constraints for ASM changes. Fig. 8(right) shows the distance to the feasible region for P-APG method, which converges to zero very fast regardless of the dimension of the problems in all of our tests. In summary, the issues with the active set method are: (i) the majority of the time is spent for identifying the optimal active set before making a noticeable progress in terms of suboptimality; (ii) as the number of active constraint in the algorithm increases, solving the KKT system in (59) becomes costly – this operation is similar to the factorization steps in interior-point methods.

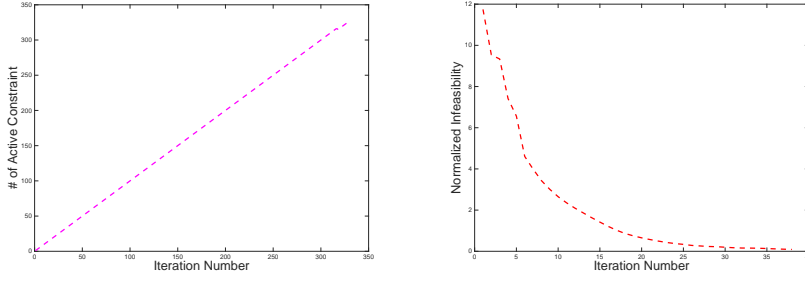


Fig. 8: P-APG vs ASM  $(n, N) = (10, 100)$ : (left) Number of Active Constraints for ASM, (right) Normalized infeasibility for P-APG

In Tables 6 and 7, besides the statistics reported in Section 4.1, we also report **Accuracy** which measures the solution quality with respect to the original problem in (3). In particular, given an approximate solution  $\tilde{\mathbf{y}}$ , obtained by solving either (3) or (7) depending on the algorithm chosen, **Accuracy** is computed as  $\|\tilde{\mathbf{y}} - \mathbf{y}^*\| / \sqrt{N}$ . As in Section 4.1, **Preprocess** for P-APG method denotes the wall-clock time used for computing the maximum singular value for matrix  $A_4$ , and **Preprocess** for ADMM accounts for  $\tilde{\Delta}_j$  computation for all  $j \in \mathcal{N}$  as shown in Figure 4. The performance comparison is shown in Table 6 and Table 7, which clearly display that as the number of observations  $N$  increases, ASM starts struggling to finish the job within 2 hours beyond  $N=800$ , and the gap between P-APG and ADMM closes rapidly, and eventually P-APG outperforms ADMM at  $N = 2400$ , which is also demonstrated in Fig. 9.

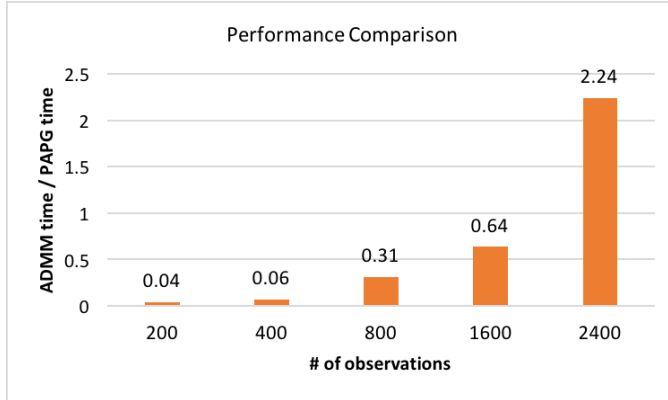


Fig. 9: Wall-time Ratio between ADMM and P-APG

## 5 Conclusion

In this paper, we proposed P-APG method to efficiently compute the least squares estimator for large-scale convex regression problems. By relaxing constraints partially, we obtained the separability on the corresponding Lagrangian dual problem. Using Tikhonov regularization, we ensured the feasibility of iterates in the limit, and we provided error bounds on 1) the distance between the inexact solution to the regularized problem and the optimal solution to the original problem, 2) the constraint violation of the regularized solution. We also proposed a continuation scheme which directly solves the (unregularized) original problem (without any negative impact on the iteration complexity), and it does not require a parameter input depending on the desired solution tolerance  $\epsilon$ . The comparison in the numerical section demonstrates the efficiency of P-APG method on memory usage compared to IPM. Furthermore, our numerical tests show that P-APG becomes the method of choice for large  $N$  values when compared to ASM and ADMM.

Table 6: Comparison of PAPG and other methods  $\frac{1}{2}\mathbf{x}^\top Q\mathbf{x}$

n, N	Algorithms	Cores/RAM	Preprocess	Wall-time	Infeasibility	Accuracy	SubOpt_Reg
80, 200	Mosek	2/2	0	5	0	0	—
	Mosek_Reg	2/2	0	9	0	1.30E-03	0
	ADMM	2/2	3	4	5.48E-04	4.32E-03	—
	ASM	2/2	0	111	0	1.31E-03	9.33E-06
	PAPG_A	2/2	1	105	7.29E-02	1.31E-03	6.54E-05
	PAPG_C	2/2	0	43	8.43E-02	1.31E-03	3.23E-06
80, 400	Mosek	4/4	0	20	0	0	—
	Mosek_Reg	4/4	0	36	0	2.60E-03	0
	ADMM	4/4	11	20	4.46E-04	4.71E-03	—
	ASM	4/4	0	665	0	2.57E-03	5.10E-07
	PAPG_A	4/4	3	319	9.38E-02	2.56E-03	1.17E-03
	PAPG_C	4/4	3	176	9.79E-02	2.56E-03	5.72E-04
80, 800	Mosek	8/8	0	109	0	0	—
	Mosek_Reg	8/8	0	188	0	3.80E-03	0
	ADMM	8/8	50	121	3.78E-04	4.98E-03	—
	ASM	8/8	0	7006	0	3.82E-03	1.47E-05
	PAPG_A	8/8	12	391	7.55E-02	3.81E-03	3.73E-04
	PAPG_C	8/8	12	281	9.89E-02	3.81E-03	2.17E-04
80, 1600	Mosek	16/16	0	544	0	0	—
	Mosek_Reg	16/16	0	918	0	2.40E-03	0
	ADMM	16/16	221	537	2.93E-04	4.96E-03	—
	ASM	16/16	0	>2 hours	N/A	N/A	N/A
	PAPG_A	16/16	50	844	9.68E-02	3.16E-03	2.22E-03
	PAPG_C	16/16	50	802	9.96E-02	3.17E-03	2.06E-03
80, 2400	Mosek	24/24	0	2537	0	0	—
	Mosek_Reg	24/24	0	4576	0	3.40E-03	0
	ADMM	24/24	678	2332	2.16E-04	4.99E-03	—
	ASM	24/24	0	>2 hours	N/A	N/A	N/A
	PAPG_A	24/24	155	1040	9.69E-02	3.26E-03	9.24E-04
	PAPG_C	24/24	155	1184	9.88E-02	3.26E-03	1.30E-03



Table 7: Comparison of PAPG and other methods  $\exp(\mathbf{p}^\top \mathbf{x})$ 

n, N	Algorithms	Cores/RAM	Preprocess	Wall-time	Infeasibility	Accuracy	SubOpt.Reg
80, 200	Mosek	2/2	0	5	0	0	—
	Mosek_Reg	2/2	0	7	0	4.41E-04	0
	ADMM	2/2	3	4	6.41E-04	4.96E-03	—
	ASM	2/2	0	155	0	4.41E-04	2.51E-05
	PAPG_A	2/2	0	64	9.21E-02	4.43E-04	6.68E-03
	PAPG_C	2/2	0	41	9.27E-02	4.42E-04	1.60E-03
80, 400	Mosek	4/4	0	19	0	0	—
	Mosek_Reg	4/4	0	33	0	9.28E-04	0
	ADMM	4/4	11	18	5.51E-04	4.95E-03	—
	ASM	4/4	0	827	0	9.29E-04	2.90E-05
	PAPG_A	4/4	3	167	9.39E-02	9.24E-04	3.00E-03
	PAPG_C	4/4	3	143	9.63E-02	9.24E-04	2.92E-03
80, 800	Mosek	8/8	0	136	0	0	—
	Mosek_Reg	8/8	0	175	0	9.74E-04	0
	ADMM	8/8	50	97	3.50E-04	4.98E-03	—
	ASM	8/8	0	> 2 hours	N/A	N/A	N/A
	PAPG_A	8/8	13	236	9.12E-02	9.72E-04	9.44E-04
	PAPG_C	8/8	13	209	9.87E-02	9.69E-04	1.65E-04
80, 1600	Mosek	16/16	0	843	0	0	—
	Mosek_Reg	16/16	0	1107	0	1.40E-03	0
	ADMM	16/16	266	787	2.66E-04	4.98E-03	—
	ASM	16/16	0	> 2 hours	N/A	N/A	N/A
	PAPG_A	16/16	62	461	9.72E-02	1.45E-03	3.07E-03
	PAPG_C	16/16	59	742	9.95E-02	1.44E-03	1.28E-04
80, 2400	Mosek	24/24	0	2522	0	0	—
	Mosek_Reg	24/24	0	3365	0	1.80E-03	0
	ADMM	24/24	846	2486	2.12E-04	4.95E-03	—
	ASM	24/24	0	> 2 hours	N/A	N/A	N/A
	PAPG_A	24/24	138	774	9.69E-02	1.82E-03	7.82E-03
	PAPG_C	24/24	139	1379	9.94E-02	1.81E-03	2.23E-03

## References

1. Aguilera, N.E., Morin, P.: Approximating optimization problems over convex functions. *Numerische Mathematik* **111**(1), 1–34 (2008)
2. Aguilera, N.E., Morin, P.: On convex functions and the finite element method. *SIAM Journal on Numerical Analysis* **47**(4), 3139–3157 (2009)
3. Aybat, N.S., Iyengar, G.: A unified approach for minimizing composite norms. *Mathematical Programming, Ser. A* **144**(1-2), 181–226 (2014)
4. Aybat, N.S., Wang, Z.: A parallel method for large scale convex regression problems. In: 53rd IEEE Conference on Decision and Control, pp. 5710–5717. IEEE (2014)
5. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.* **2**(1), 183–202 (2009)
6. Bertsekas, D.P.: *Nonlinear Programming*. Athena Scientific (1999)
7. Birke, M., Dette, H.: Estimating a convex function in nonparametric regression. *Scandinavian Journal of Statistics* **34**(2), 384–404 (2007)
8. Boyd, S., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press, New York, NY, USA (2004)
9. Chen, C., He, B., Ye, Y., Yuan, X.: The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming* **155**(1-2), 57–79 (2016)
10. Chen, H., Yao, D.: Fundamentals of queueing networks: Performance, asymptotics, and optimization, *Stochastic Modelling and Applied Probability*, vol. 46. Springer (2001)
11. Dent, W.: Note-a note on least squares fitting of functions constrained to be either nonnegative, nondecreasing or convex. *Management Science* **20**(1), 130–132 (1973)

12. Engl, H., Kunisch, K., Neubauer, A.: Convergence rates for Tikhonov regularisation of non-linear ill-posed problems. *Inverse problems* **5**(4), 523 (1989)
13. Gill, P.E., Golub, G.H., Murray, W., Saunders, M.A.: Methods for modifying matrix factorizations. *Mathematics of Computation* **28**(126), 505–535 (1974)
14. Groeneboom, P., Jongbloed, G., Wellner, J.: Estimation of a convex function: Characterizations and asymptotic theory. *Annals of Statistics* **29**(6), 1653–1698 (2001)
15. Hannah, L., Dunson, D.: Approximate dynamic programming for storage problems. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 337–344 (2011)
16. Hannah, L., Dunson, D.: Ensemble methods for convex regression with applications to geometric programming based circuit design. *arXiv preprint arXiv:1206.4645* (2012)
17. Hannah, L., Dunson, D.: Multivariate convex regression with adaptive partitioning. *The Journal of Machine Learning Research* **14**(1), 3261–3294 (2013)
18. Hanson, D., Pledger, G.: Consistency in concave regression. *The Annals of Statistics* pp. 1038–1050 (1976)
19. Hildreth, C.: Point estimates of ordinates of concave functions. *Journal of the American Statistical Association* **49**(267), 598–619 (1954)
20. Holloway, C.A.: Technical note on the estimation of convex functions. *Operations Research* **27**(2), 401–407 (1979)
21. Kuosmanen, T.: Representation theorem for convex nonparametric least squares. *The Econometrics Journal* **11**(2), 308–325 (2008)
22. Lim, E., Glynn, P.: Consistency of multidimensional convex regression. *Operations Research* **60**(1), 196–208 (2012)
23. Magnani, A., Boyd, S.P.: Convex piecewise-linear fitting. *Optimization and Engineering* **10**(1), 1–17 (2009)
24. Mammen, E.: Nonparametric regression under qualitative smoothness assumptions. *The Annals of Statistics* pp. 741–759 (1991)
25. Mazumder, R., Choudhury, A., Iyengar, G., Sen, B.: A computational framework for multivariate convex regression and its variants. *arXiv preprint arXiv:1509.08165* (2015)
26. Meyer, R., Pratt, J.: The consistent assessment and fairing of preference functions. *Systems Science and Cybernetics, IEEE Transactions on* **4**(3), 270–278 (1968)
27. MOSEK: The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 53). (2016). URL <http://docs.mosek.com/7.1/toolbox/index.html>
28. Mousavi, M., Glynn, P.: Shape-constrained estimation of value functions. preprint available at [arXiv:1312.7035](http://arxiv.org/abs/1312.7035) (2013)
29. Nesterov, Y.: Introductory lectures on convex optimization, *Applied Optimization*, vol. 87. Kluwer Academic Publishers, Boston, MA (2004)
30. Nesterov, Y.: Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization* **16**(1), 235–249 (2005)
31. Nesterov, Y.: Smooth minimization of nonsmooth functions. *Mathematical Programming, Series A* **103**, 127–152 (2005)
32. Nocedal, J., Wright, S.J.: *Numerical Optimization*, 2nd edn. Springer, New York (2006)
33. Nocedal, J., Wright, S.J.: *Numerical Optimization* (2nd edition). Springer (2006)
34. Schmidt, M.W., Le Roux, N., Bach, F.: Convergence rates of inexact proximal-gradient methods for convex optimization. In: *NIPS*, pp. 1458–1466 (2011)
35. Seijo, E., Sen, B.: Nonparametric least squares estimation of a multivariate convex regression function. *The Annals of Statistics* **39**(3), 1633–1657 (2011)
36. Shively, T.S., Walker, S.G., Damien, P.: Nonparametric function estimation subject to monotonicity, convexity and other shape constraints. *Journal of Econometrics* **161**(2), 166–181 (2011)
37. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization (2008). Preprint available at <http://www.eecs.berkeley.edu/~brecht/eecs227cdocs/tseng.pdf>
38. Yen, N.D.: Lipschitz continuity of solutions of variational inequalities with a parametric polyhedral constraint. *Mathematics of Operations Research* **20**(3), 695–708 (1995)